The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

NOVEMBER/DECEMBER 1988
Volume 6.6
$5.00 Canada $6.25

Numerics

Blazing Speed on the PC

Ray Duncan on Forth

Graphics

Saving VGA Screens

OS/2 & Unix

Ada Reliability

DeSmet C & the ANSI Standard

21609 PJS*****2MARAPR89
RON K UNZ
WALL STREET ANALYTICAL INC
33-41 85TH ST
JACKSON HEIGHTS NY 11372

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Make your programs millions of times smarter.

More and more, programmers and workstation builders are using DESQview 2.0 as a development tool. The reason is simple. They can create powerful, multitasking solutions today for the millions of DOS PCs in use today. Solutions comparable to those promised for tomorrow by OS/2.

Some of the applications under development right now using DESQview 2.0 API Tools: CAD, medical systems, insurance, 3270 mainframe communications, network management, real estate, typesetting, point of sale, education, commodity trading, stock trading and online voting.

The API Advantage

Programmers who take advantage of DESQview's API (Application Program Interface) get access to the powerful capabilities built into DESQview--multitasking, windowing, intertask comunications, mailboxes, shared programs, memory management, mousing, data transfer, menu-building and context sensitive help.

Bells and Whistles

A program taking advantage of the DESQview 2.0 API can spawn subtasks for performing background operations or new processes for loading and running other programs concurrently. It can schedule processing after an interval or at a certain time. It can use DESQview's intertask communications to rapidly exchange data between programs, share common code and data, or interrupt at critical events. It can use DESQview's menuing and mousing capabilities to create menus. And there's lots more it can do.

80386 Power

80386 programmers can take advantage of the 80386's protected mode for large programs, yet run on DOS and multitask in DESQview--side by side with other 80386 and DOS programs. The breakthroughs that make this possible: DOS Extenders from PharLap Software and AI Architects and DESQview support of these DOS extenders.

DESQview Developer Support

So if you are a developer, looking to create programs with mainframe capabilities, but wanting to sell into the existing base of millions of DOS PCs, we can be very helpful. We offer DESQview API Tools for assembler or C programmers: a debugger, a Reference Manual. And lots more tools on the way.

Plus, we offer DESQview API Developers Conferences. Come learn about the DESQview 2.0 API and 80386 DOS Extenders. Meet 80386 experts as well as those smart people who are creating DESQview 2.0 API workstation solutions.

For more information, call or write us.

Bringing New Power to DOS. DESQview 2.0 API Toolkit.

Quarterdeck Office Systems 150 Pico Blvd.,Santa Monica, CA 90405
(213) 392 9851

Number 147 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Simply the BEST C and Pascal on AT, 386, Sun, Apollo, RT, VAX, 370

"The most rock-solid C compiler in the industry. Superb technical support and portability. Superior code generated."
Gordon Eubanks, Symantec — Q&A (386).

"It simply works, with no trouble, no chasing strange bugs, and excellent warning and error messages ... a professional product."
Robert Lerche, Bay Partners.

"For large-scale software development, the highest quality C compiler available on the market today. Pragmas are great. Quality of support is exceptional." **Randy Neilsen, Ansa—Paradox (DOS,OS/2).**

"15% smaller and 15% faster than Lattice C."
Robert Wenig, Autodesk.

"Our software is running anywhere from 30 to 50% faster than when compiled under Lattice." **David Marcus, Micronetics.**

"We switched from Lattice due to a 10% reduction in code size. The compiler is very stable." **Lee Lorenzen, Ventura Software — Ventura Publisher, marketted by Xerox Corp.**

"Best quality emitted code by any compiler I've encountered. Often amazing." **Bill Ferguson, Fox Software — FoxBase (386).**

"Messages sometimes pointed out type mismatches, incorrect-length argument lists, and uninitialized variables that had been undetected for years [in 4.x bsd]." **Larry Breed, IBM ACIS [RT PC].**

"Diagnostics turned up bugs missed by other compilers. Rapid bug fixes by technical support, someone who knew what he was talking about. 80386 code is well optimized."
Tim Addison, Logistics Data Systems.

"386 protected mode support is fantastic, especially the access to large amounts of memory. It's mainframe compute power on a PC." **Dan Eggleston, Viewlogic.**

"The preprocessor supplied with Professional Pascal is quite useful. The code quality and control over segmentation and memory models are superior to MS Pascal." **Bob Wallace, QuickSoft.**

Check Out These Reviews

- **High C ™:**

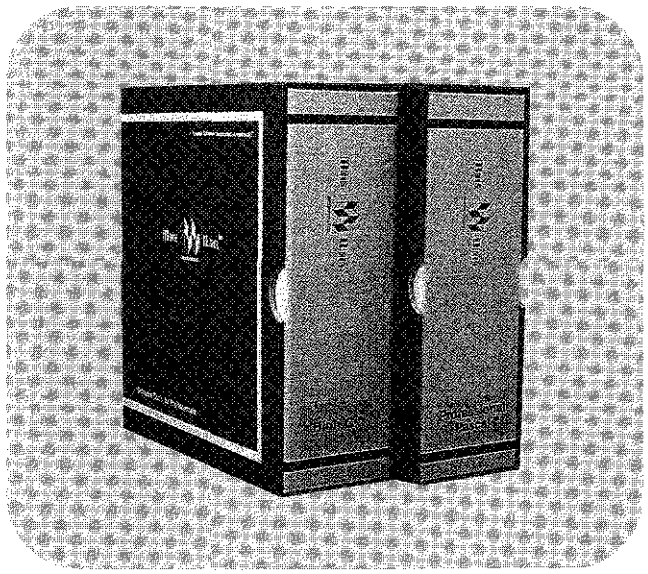
Computer Language	February 1986, '87	
Dr. Dobb's Journal	August 1986	
PC Magazine	Jan. 27, 1987	(80386 version)
Dr. Dobb's Journal	July 1987	(80386 version)
BYTE Magazine	November 1987	(80386 version)

- **Professional Pascal ™:**

PC Magazine	Dec. 29, 1985	
Computer Language	May 1986	
PC Tech Journal	July 1986	
Journal of Pascal, Ada, & Modula-2	Nov.-Dec. 1986	
BYTE Magazine	Dec. '86, June '87	(80386 version)

Why MetaWare compilers

- They are specifically designed for serious software developers.
- They are reliable and robust: they don't break at every turn.
- Their generated code is the best, or near best, on each architecture.
- Their superior diagnostic messages help you produce better products more quickly.
- Your source can be ported with ease to the most popular systems.
- You can link mixed-language modules from our compilers, others
- You can benefit from high-level, personal technical support.
- You can take advantage of the latest ANSI C extensions, and/or extensive Pascal extensions. **High C** has been tracking the ANSI Standard for two years; **Professional Pascal** will soon have a VS Pascal compatibility switch and several Apollo Pascal ext'ns.
- You can take advantage of the latest 387 and Weitek 1167 support — we have the only compilers with Weitek real mode support.



Power Tools for Power Users

Ashton-Tate: dBase III Plus, MultiMate; Autodesk: AUTOCAD, AUTOSKETCH (8087, '387, Weitek); Boeing Computer Services (Sun); CASE Technology (Sun); CAD/CAM giant Daisy ('86, '386, VAX); Deloitte Haskins & Sells; Digital Research: FlexOS; GE; IBM: 4.3/RT, 4680 OS; Lifetree Software (Pascal): Volkswriter Deluxe, GEM-Write; Lugaru: Epsilon; NYU: Ada-Ed cmplr; Semantec: Q&A; Sky Computers; ... (Product names are trademarks of the companies indicated.)

Industrial-Strength

MetaWare C and Pascal compilers are designed for professional software developers. These tools are loaded with options to control them for special purposes. You can adjust the space-time trade-off in code quality. You can adjust external naming conventions to agree with linkers and operating systems. You can specify segment names for segmented architectures, and to help place code or data in particular places for embedded applications. You can select from five memory models for the 8086 family. And on and on.

A Partial List of Optimizations

Common subexpression and dead-code elimination, retention and reuse of register contents, jump-instruction size minimization, tail merging (cross jumping), constant folding, short-circuit evaluation of Boolean expressions, strength reductions, fast procedure calls, automatic mapping of variables to registers (where advantageous), ...

"Platform" — Code Quality

Sun,Apollo,SGI	— 18%, 3%, 26% > resident compiler (Dhrystone).
PC: DOS, OS/2	— 3-10% > Microsoft C; 30% > MS Pascal, LatticeC.
386 32-bit DOS	— no competitors, since November, 1986.
286, 386 UNIX	— 66% better than pcc (Dhrystone, 386).
VAX VMS	— ≈ DEC's excellent C and Pascal; better features.
VAX Ultrix	— 19% > pcc (Dhrystone); much > Berkeley Pascal.
RT PC/4.3bsd	— 89% > the original port of pcc (Dhrystone).
370 CMS,UNIX	— much better than any C, and VS Pascal.
AMD 29000	— >40,000 Dhrystones! Available in Q2, cross.

(408) 429-6382, telex 493-0879. ***Since 1979.***

MetaWare™ INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

The Clear Choice for Large Programming Projects — PC Tech J.

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others and their owners are duely respected.

Number 45 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

Volume 6.6
November/December 1988

Contents

4 Notes from the Editors

6 Letters
Michael Burton, Michael Frame & Tom Swan, W.G. Madison & Tom Swan, Larry Smith, Terry Ritter

19 PJ Forum: DOS 4.0 Leaps into the OS Mix *by William G. Madison*
Market pressure exerts its influence

22 Redirections: OS/2 vs. Unix *by Frank D. Greco*
Our Unix correspondent compares the rival multitaskers

32 On Graphics: Saving VGA Screens, Color Selection, and Other Mysteries *by Michael Abrash*
The less obvious details and MASM quirks revealed

45 DOS: 64K Extra Memory for DOS Applications *by Jake Richter*
Where it comes from and how to address it

46 C: DOS Device Drivers—A Debugging and Information Utility *by Stephen A. Rodgers*
Examine device drivers

53 C: The Proposed ANSI C Language Standard *by Rex Jaeschke*
Rex looks at DeSmet C V3.1

56 Forth: A Forth Apologia *by Ray Duncan*
With no apologies

64 Ada: Programming for Reliability *by Richard D. Riehle*
Data Types and Exceptions

72 Numerics: Crunching Numbers at Blazing Speed *by M. Steven Baker*
The PC invades the last preserve of minis and mainframes

90 Miscellaneous *by M. Steven Baker*
Make utilities (PolyMake & Seidel), text editors and formatters (SoftQuad & DeSmet's SEE), programming utilities (C-LINES/C-TREE), multitaskers (DESQview)

95 PJ Billboard

96 Advertiser Index

The ancient abacus was a major advance in number crunching. New math coprocessors for the PC (the Weitek "Abacus," for instance) provide the next performance advance in numerics. *Artist Ralph Bentley*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SERIOUS DEBUGGING *at a* REASONABLE PRICE

All the speed and power of a hardware-assisted debugger at a software price

Soft-ICE

Hardware-level break points

REAL-TIME break points on memory locations, memory ranges, execution, I/O ports, hardware and software interrupts. More powerful break points than ANY software-only debugger on the market. Soft-ICE gives you the power of an in-circuit emulator on your desk.

Break out of hung programs

With a keystroke - no external switch necessary. Even with interrupts disabled.

Breaks the 640K barrier

Soft-ICE uses ZERO bytes of memory in the first 1MB of address space. This is especially useful for those subtle bugs that change when the starting address of your code changes. With Soft-ICE your code executes at the same address whether the debugger is loaded or not.

Works with your favorite debugger

Soft-ICE can be used as a stand-alone debugger or it can add its powerful break points to the software debugger you already use. You can continue to use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated.

Solve tough systems problems too

Soft-ICE is ideal for debugging TSRs, interrupt handlers, self booting programs, DOS loadable device drivers, non-DOS operating systems, and debugging within DOS & BIOS. Soft-ICE is also great for firmware development because Soft-ICE's break points work in ROM.

How Soft-ICE Works

Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to provide real-time hardware-level break points.

"Soft-ICE is a product any MS-DOS developer serious enough to own a 386 machine should have."

Dr. Dobb's Journal — May 1988

RUN CODEVIEW IN ONLY 8K!

MagicCV

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 microprocessor to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView using less than 8K of conventional memory on your 80386 PC.

Don't let 640K be your limit!

If you are closing in on the 640K limit and would like the power of CodeView, MagicCV is for you.

Don't let the debugger hide the bug!

Even if you're not closing in on the 640K limit, running CodeView with MagicCV makes your debugging environment much closer to the end user's program environment. You can use CodeView to locate subtle bugs that only occur when there is plenty of free memory, or those difficult bugs that only occur when your program is running with a couple of TSRs loaded.

How MagicCV works

MagicCV uses the 80386 to create a separate virtual machine for CodeView. MagicCV uses between 4K & 8K of conventional memory as a bridge between the DOS environment and CodeView.

MagicCV is easy to use

If you are a CodeView user, you already know how to use MagicCV too. Just type MCV instead of CV; everything else is automatic.

Save $86

MagicCV	$199
Soft-ICE	$386

Buy Both and Save $86!

CALL TODAY
(603) 888 - 2386
***or FAX* (603) 888 - 2465**

30 day money-back guarantee
Visa, Master Card and AmEx accepted

NU-MEGA TECHNOLOGIES

P.O. BOX 7607 • NASHUA, NH 03060-7607

MagicCV with Soft-ICE

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

Both require 80386 AT compatible or IBM PS/2 Model 80. MagicCV requires at least 384K of extended memory. CodeView is a trademark of Microsoft Corporation.

Number **160** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers
PROGRAMMER'S Journal

NOTES from the Editors

Editor
Erin O'Connor
Technical Editor
M. Steven Baker
Contributing Editors
Michael Abrash, Frank D. Greco, Tom Swan

Contributing Writers
Michael Abrash
M. Steven Baker
Ray Duncan
Frank D. Greco
Rex Jaeschke
W.G. Madison
Jake Richter
Richard D. Riehle
Stephen A. Rodgers

Publisher
Elizabeth Oakley
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Andrea McHone
Subscriptions
Gary Hoyer
Business Manager
Bobbi Sinyard
Administrative Assistant
Lori Fanslow

Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
(503) 747-0800

Subscriptions: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
Change of Address
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.

Code and listings in Programmer's Journal appear "as is," and Programmer's Journal makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. Programmer's Journal assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan or Andrea McHone (503/747-0800), or write to Programmer's Journal, P. O. Box 30160, Eugene, OR 97403.

Programmer's Journal is produced on Ventura Publisher, a product of Xerox Corporation.

Programmer's Journal (ISSN 0747-5861) is published bimonthly by Oakley Publishing Company. Second-class postage paid at Springfield, OR, and additional mailing offices.

POSTMASTER: Send address changes to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

© 1988 OAKLEY PUBLISHING COMPANY

In the midst of his long running series on the ANSI C Standard, Rex Jaeschke pauses in this issue to entertain an heretical notion. Strict conformance to the ANSI Standard isn't always the sine qua non of excellence. Ray Duncan's known that for a long time, but he seems to feel that an ANSI conforming Forth isn't a bad idea, for a lot of reasons. Forth's impending standardization is a good occasion for a Forth piece, and in this issue Ray disabuses us of some of our most cherished prejudices.

Year-end Report

We've been chasing the 80386 story for the last year and a half. Early on, Steve Baker saw possibilities in the native mode of the 386 and got busy. Steve assembled his tools in "Porting Mainframe Programs" in Volume 6.1 and in Volume 6.2's "80386 Development Tools." Now Steve's realizing the fruits of his investigations and exertions. In this issue Steve brings the DOS extenders, the 386 FORTRAN and C compilers, the Definicon Hummingboard, the 387 and Weitek math chips, and a couple Compaqs to bear on some impressive results for numerics intensive applications on the 386. In an upcoming issue, Steve will follow up his X Window story.

Cut Short

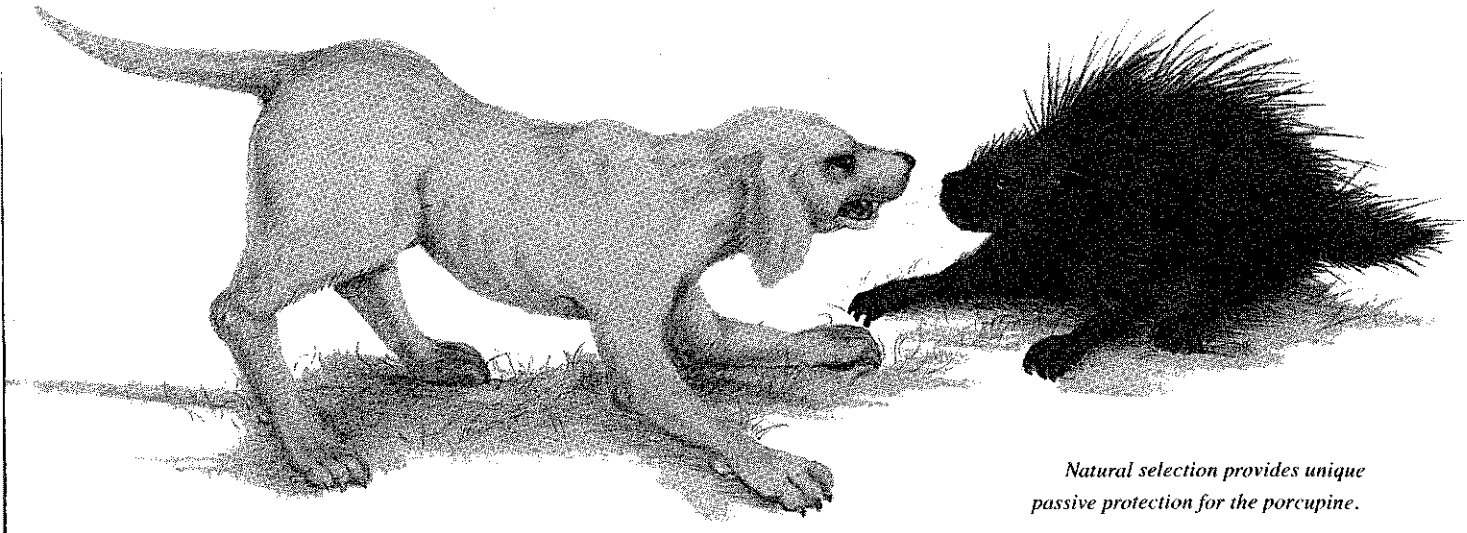
Here at *PJ* we've already experienced the effects of Borland's efforts to wring a bigger profit out of its huge sales volume. We lost Borland's back cover ad, a long-standing arrangement that in the early years of *PJ* was more a pat on the back for us than a money-maker for Borland. Along about the same time, familiar personalities from the various Borland departments disappeared. The next time we looked up, *Turbo Technix* was gone. *Turbo Technix* was a cut above the usual house organ. Working with a palette limited to Borland products, Jeff Duntemann & Company put together a lively, literate, and technically impeccable bimonthly publication for "beginners," "programmers," and "wizards." They put out six good issues, and I hear the seventh was to be even better. I'm sorry we won't get to see that last issue.—*Erin*

Microsoft has been king of the mountain for some time now, with a complete line of language products—macro assembler, linker, librarian, their CodeView symbolic debugger, and various compilers. Since May, most of these products have been available under OS/2. And although Microsoft's other language compilers are weaker, Microsoft C is considered the DOS compiler to beat. Yes, competitors against individual Microsoft language tools have been introduced this year—OPTASM (from SLR systems) and Watcom C and Zortech C++. And significant updates include Borland's Turbo C, MetaWare's High C, Aztec C from Manx, Lattice C, Computer Innovations' C-86, C Ware's DeSmet DC88, and Phar Lap's tools. Powerful debuggers are available from NuMega, Periscope, and Atron. But no one firm has offered a line of products that directly competes with and replaces Microsoft's language tools. This is about to change.

In March I spoke with Borland about their plans for new language products. They projected early summer for the release of their debugger and late summer for OS/2 language products. I sensed then that their less profitable products (the toolboxes) would be sold to other companies, and they've already let their Modula-2 compiler go to Jensen. The promised language products have been slow to emanate from Borland, and I'm still wondering how far their focus has changed to business applications—Quattro, Sprint, and Paradox. Some faces have changed at Borland, too, and they are certainly more corporate and profit oriented than they've been in the past. But with recent announcements, Borland clearly has the capability to compete directly against most of the entire Microsoft language development family.

Turbo Prolog 2.0 was followed quickly by Turbo C 2.0, Turbo Pascal 5.0, and Borland's macro assembler and debugger (TASM). These are strategic products for Borland, not because of the potential market for another assembler or debugger, but because Borland now has a complete line of language development tools. Like other vendors, Borland has always had to play catchup with Microsoft and support this or that new feature (CodeView, for example). Now, if they can maintain their momentum, Borland can begin to lead, not follow. The symbolic debugger is the key product that adds value to the other tools. The debugger is equal to, if not a significant enhancement over, CodeView. On 80386 machines, the debugger can be run in a virtual 8086 machine, leaving a full 640K for DOS applications. The remote capabilities of the Borland debugger leave CodeView in the dust. And most important, the debugger has a well documented interface allowing other products (hardware assisted debuggers) to communicate gracefully. Because of this interface, converting the debugger to OS/2 shouldn't be a problem. And porting the Borland compilers to OS/2 shouldn't be that hard. The only difficult part will be rewriting the runtime libraries, the same task Microsoft faces for their language compilers except C—the Microsoft C runtime fully supports OS/2. The question remains, though, whether Borland is fully committed to their language development tools or is just finishing up a product development cycle.—*Steve*♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Natural selection provides unique passive protection for the porcupine.

The Activator - Natural Selection For Software Protection

Inventor and entrepreneur Dick Erett explains how "The Activator" provides sane protection for your intellectual property.

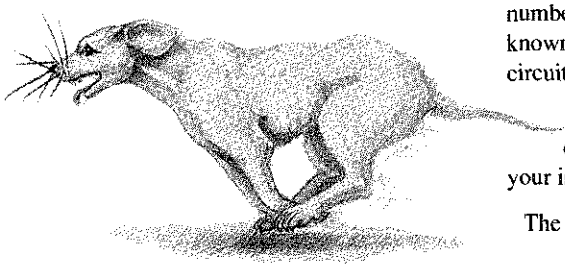
"In any industry, just as in nature, the process of natural selection raises one solution above another. Natural selection is the most elegant of engineers.

In the area of software protection The Block has been selected by the marketplace as the solution that works. Over 500,000 packages are protected by our device.

For the past 4 years our philosophy has been; *'You have the right and obligation to protect your intellectual property.'*

A New Ethic For Software Protection

In allowing end-users unlimited copies of a software package and uninhibited hard disk and LAN operation, The Block has created a new ethic for software protection.



By removing protection from the magnetic media we remove the constraints that have plagued legitimate users.

They simply attach our key to the parallel port and forget it. It is totally transparent, but the software will not run without it.

A New Technology For Software Protection

Our newest model, The Activator, builds on our current patented design, and establishes an unprecedented class of software protection.

We have migrated and enhanced the circuitry of The Block to an ASIC (Application-Specific Integrated Circuit) imbedded in The Activator.

This greatly improves speed and performance, while reducing overall size. Data protection can also be provided.

Programmable Option

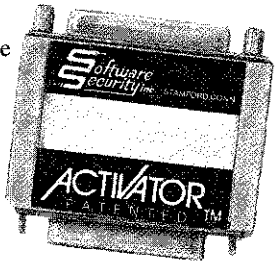
The Activator allows the software **developer** the option to program serial numbers, versions, or other pertinent data known only to the developer, into the circuit, and access it from the program.

Once you program your part of the chip, even we have no way to access your information.

The ASIC makes emulation of the device virtually impossible. It also presents an astronomical number of access combinations.

Full 100% Disclosure

Since The Activator is protected by our patent we fully disclose how it works. Once you understand it, endless methods of protection become evident.



Just as no two snowflakes are the same, no two implementations of The Activator are identical. And like the snowflake the simplicity of The Activator is its greatest beauty.

We never cramp your programming style or ingenuity. Make it as simple or complicated as you desire.

Let us help safeguard what's rightfully yours. Please call today for additional information or a demo unit. *It's only natural to protect your software."*

1-800-333-0407 ext.103
In Connecticut 203-329-8870
Fax 203-329-7428

Software Security inc.

870 High Ridge Road
Stamford, CT 06905

Unlimited Copies • Programmable • No Batteries • Small Size • Fast • Patented • Data Protection

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If you can find better sort/merge software, buy it!

Sortex:® A flexible, general purpose sort/merge facility which provides you with the ultimate in performance and reliability

- Runs from the DOS prompt, batch file, or applications written in Assembler, BASIC, COBOL, FORTRAN, Pascal, or C.
- Accepts your requirements through the full screen menu/panel facility, control file, or command line.
- Processes most of the commonly used file and data types.
- Produces output files with key, address, index, tag, or full records.
- Imposes no practical limitations. Files with four billion characters, records & fields with thirty two thousand characters each, and unlimited number of keys can be handled by Sortex.
- Allows records to be chosen selectively for processing.
- Sorts your records in ascending/descending order on any number of keys.
- Merges your sorted record sequences into one sequence on any number of keys.
- Preserves the input order of records on the output file if all their keys are identical.
- Allows keys to overlap, or be contained within other keys. They need not be contiguous.
- Processes files that contain both standard and nonstandard delimiters.

30-Day Money-Back Guarantee
Only $149.95

CORPORATION
1915 W. Orangewood Avenue
Orange, California 92668
Tel.: 714 978 0480

Number 171 On the Reader Service Card

Mouse Update

After seven months of intensive mouse programming (I'm beginning to see mice in my sleep), I was pleased to see John J. Newlin's "Not a Creature Was Stirring" in Volume 6.4. I'd like to clarify and add to a few of Mr. Newlin's points. My comments are based on Version 6.11 of the Microsoft mouse driver.

Mr. Newlin states that the high resolution mode of the CGA is 600 x 200. It is actually 640 x 200, and the mouse reports its cursor position within these 0–640 and 0–200 limits. It is the responsibility of the mouse programmer to scale the mouse cursor position reports to the video mode currently selected.

In the explanation of mouse function 12, a table of triggers is shown. The implication is that only one of these bits may trigger the interrupt handler. Actually, any number of these bits may be set. For instance, if you wish to be interrupted on any mouse movement or button release, the value for CX would be 1 + 4 + 16 = 21. It is then the mouse interrupt handler's responsibility to determine what actually caused the interrupt.

The article states that when a mouse event that activates the mouse interrupt handler occurs, the mouse driver supplies certain information to your handler. The information is placed in AX, BX, CX, and DX. Actually, two other items are also supplied. The DI register contains the horizontal motion counter (see function 11), and the SI register contains the vertical motion counter. The motion counters are not reset as they are when calling function 11.

Mr Newlin has also apparently been working with an older version of the Microsoft mouse driver and their *Mouse Programmer's Reference Guide*. Several new functions have been added, which are documented in the latest *Mouse Programmer's Reference Guide*. The new functions are explained below.

Mouse function 16 allows you to define a region on the display where the mouse cursor will automatically be hidden. If the mouse cursor is within this region when the function call is made, or if it enters the region after the call is made, it will be hidden. A call to function 1 is necessary to restore the cursor. Function 16 requires AX = 16, CX = the starting horizontal screen coordinate, DX = the starting vertical screen coordinate, SI = the ending horizontal screen coordinate, and DI = the ending vertical screen coordinate.

Mouse function 20 allows you to exchange the current mouse interrupt handler (enabled by function 12) with another interrupt handler. This can be very handy for TSRs that require use of the mouse and then wish to return mouse control to whatever they interrupted. Function 20 requires AX = 20, CX = the new trigger, and the ES:DX register pair pointing to your new mouse interrupt handler. Function 20 returns the old trigger in CX and the old mouse interrupt handler address in ES:DX.

Mouse function 21 allows you to determine how much memory is required to save all the information about the current state of the mouse driver. Function 21 is used in conjunction with functions 22 and 23, explained below. Function 21 requires AX = 21, and it returns the size of the buffer (in bytes) required to save the mouse driver state in BX.

Mouse function 22 saves the mouse driver state by copying the internal mouse driver state variables to a buffer area in your program. The variables can be restored using function 23. Function 22 requires AX = 22 and the register pair ES:DX to point at the buffer where the mouse state is to be saved.

Mouse function 23 restores the internal mouse driver state variables from an external buffer where they have been saved. Function 23 requires AX = 23 and the register pair ES:DX to point at the buffer where the mouse state has been saved.

Mouse function 29 switches the video display page to a new page and tells the mouse driver that the new page is where the mouse cursor will be displayed. Obviously, this works only if your display adapter has enough memory to display more than one page. Function 29 requires AX = 29 and BX = the video ram page number, counting from zero.

Mouse function 30 returns the video page number where the mouse cursor is being displayed. Function 30 requires AX = 30 and returns the video ram page number in BX.

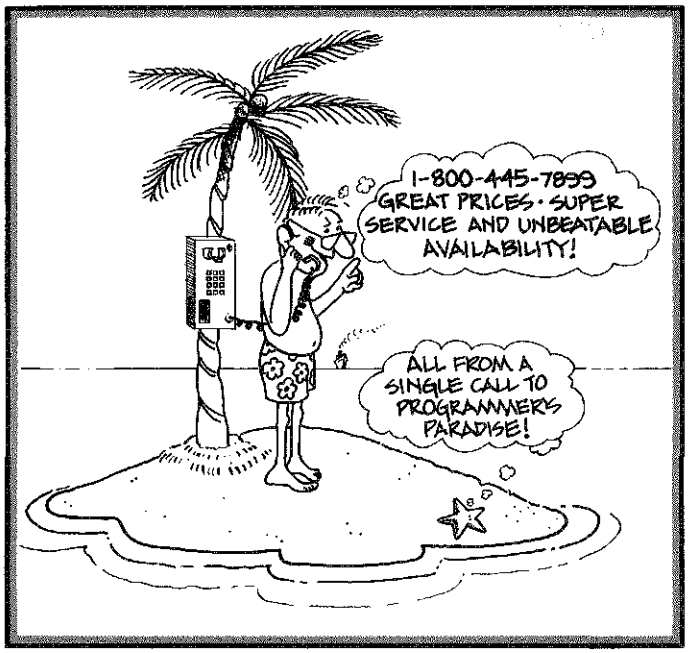
The following mouse functions are not documented in the version of the Microsoft *Mouse Programmer's Reference Guide* that I have, but this doesn't necessarily mean that they are "undocumented" functions. In using these functions, take care.

Mouse function 24 allows you to define alternate mouse interrupt handlers for mouse events that occur while the Shift, Control, or Alternate key is being pressed. This is actually an extension of function 12 capability. The trigger byte looks the same except that three new bits are added—bit 5 for the Shift key (CX = 32), bit 6 for the Control key (CX = 64), and bit 7 for the Alternate key (CX = 128). The driver maintains a table of these interrupt handlers, one entry for each com-

Continued on page 10

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PARADISE PRICES 1-800-445-7899



CALL PROGRAMMERS PARADISE TODAY **and discover our commitment to bringing you the best software at the best prices. You'll also find software pros to help you select the product that you need plus immediate shipment on our stock of 1000 products and a 30-day money back guarantee. Paradise is ready and waiting for you.**

	LIST	OURS
ASSEMBLY LANGUAGE		
Advantage Disassembler	295	279
MS Macro Assembler	150	105
OPTASM	129	115
Phar Lap 386 ASM/LINK	495	409
Turbo Assembler/Debbuger	150	105
BASIC		
MS BASIC/6.0	295	199
MS QuickBASIC	99	69
Turbo Basic	100	69
Turbo Basic Toolboxes	100	69
C COMPILERS		
High C 386	895	799
Lattice C	450	289
Microsoft C	450	299
NDP C-386	595	529
QuickC	99	69
w/serial mouse	249	149
Turbo C v. 2.0	150	105
WATCOM C 6.5	295	269
C LIBRARIES/UTILITIES		
C ASYNCH MANAGER	175	137
C TOOLS PLUS/5.0	129	101
C Utility Library	199	139
Essential Comm Library	185	125
Greenleaf Business MathLib	239	159
Greenleaf Comm Library	225	169
Greenleaf Functions	209	155
Greenleaf SuperFunctions	265	179
Greenleaf TurboFunctions	109	79
PC-lint	139	101
PforCe	395	215
TimeSlicer	295	279
Turbo C Tools	129	99
C GRAPHICS		
GraphiC	395	322
GSS Graphic Dev. Tk.	495	409
HALO '88	325	219
HALO '88-MS Developers	595	399
MetaWINDOW	195	162
MetaWINDOW/PLUS	275	232
Turbo Geometry Lib.	150	135
C SCREENS/WINDOWS		
C-Scape	299	282
JAM	750	684
PANEL Plus	495	395
PANEL/QC or TC	129	99
Vitamin C	225	162
VCScreen	100	80
Windows for C	195	169
Windows for Data	295	259
dBASE LANGUAGE		
Clipper	695	439
dBFast	100	75
FoxBASE +	395	249
Flipper	195	182
MAGIC PC	199	179
R&R	149	129
SilverComm Library	150	139
Tom Rettig's Library	100	80
FORTRAN COMPILERS		
Lahey F77 FORTRAN	477	429
MS FORTRAN	450	299
NDP FORTRAN-386	595	479
RM/FORTRAN	595	479
PASCAL COMPILERS		
Microsoft Pascal	300	199
Turbo Pascal v. 5.0	150	105
Turbo Pascal Dev. Toolkit	395	289

NEW RELEASES

ACTOR 1.2
Now supports 500K memory under LIM Expanded Memory Specifications, and adds full support for the new releases of Microsoft Windows/286 and Windows/386 (versions 2.1). List: $495 **Ours: $423**

LATTICE C 3.3
Integrated environment that includes a screen editor and source level debugger. Lattice C 3.3 generates DOS, OS/2 or 'Family mode' programs which are compatible with either operating system. List: $450 **Ours: $289**

Essential C Utility Library 4.0
New Power Functions include a fully functional editor, directory backup/maintenance utilities, mouse-driven user interfaces, dialog boxes and much more. List: $199 **Ours: $139**

	LIST	OURS
TURBO PASCAL ADD-ONS		
DATABOSS APPLIC. GEN.	399	359
T-Debug Plus v. 4.0	45	40
w/Source	90	80
Turbo Analyst	75	69
Turbo Halo	95	80
Turbo POWER SCREEN	129	99
Turbo POWER TOOLS PLUS	129	99
Turbo POWER UTILITIES	95	79
Turbo Professional 4.0	99	80
DISK/DOS/UTILITIES		
386-to-the-Max	75	66
Advanced Norton Utilities	150	99
Mace Utilities	99	90
Norton Commander	75	55
Norton Utilities	100	61
PC Tools Deluxe	80	70
Vfeature Deluxe	120	111
X-Tree Pro	129	111
EDITORS		
BRIEF	195	CALL
Epsilon	195	151
KEDIT	150	120
MKS/VI	75	66
Multi-Edit	99	90
PI Editor	195	165
SPF/PC	245	185
VEDIT PLUS 3.0	185	131
FILE MANAGEMENT		
Btrieve	245	185
Xtrieve	245	189
Report Option	145	109
Btrieve/N	595	455
c-tree	395	318
d-tree	495	395
r-tree	295	241
c-tree/r-tree Bundle	650	523
CQL Query System	395	332
dBC III	250	169
dBC III PLUS	750	599
dB_VISTA	195	159
XQL	795	599

1000 BRAND NAME PRODUCTS IN STOCK

	LIST	OURS
OBJECT-ORIENTED PROGRAMMING		
ACTOR	495	423
ADVANTAGE C++	495	479
C-talk	150	137
Smalltalk/V	100	85
Communications	50	45
EGA/VGA Color Ext.	50	45
Goodies #1 or #2	50	45
Smalltalk/V 286	200	169
OPERATING SYSTEMS/ CONTROL PROGRAMS		
Concurrent DOS 386	395	349
Microport Sys. V/AT (comp)	649	549
Sys. V/386 (comp.)	899	769
SCO XENIX Sys. V (comp)	1295	999
System V/386 (comp.)	1495	1195
for IBM PS/2 Model 80	1695	1359
VM/386	245	182
MICROSOFT Windows/386	195	130
Wendin-DOS	139	109

	LIST	OURS
OTHER PRODUCTS		
INTERACTIVE EASYFLOW	150	125
LINK & LOCATE++	395	349
Logitech MOD II Dev. Sys.	249	209
MKS Toolkit	169	145
MS COBOL Compiler	900	599
MS Windows Dev. Kit	500	319
Norton Guides	100	75
Opt-Tech Sort	149	99
PC Scheme	95	86
Periscope I,II,III	CALL	CALL
Plink86plus	495	275
PVCS Corporate	395	359
Source Print	97	80

HOW WE WORK

PHONE ORDERS
Hours 9 AM – 7 PM EST. We accept MasterCard, Visa, American Express. Include $3.95 per item for shipping and handling. All shipments by UPS ground. Rush service available, ask for rates when you order.

MAIL ORDERS
POs by mail or fax are welcome. Please include phone number.

TECHNICAL SUPPORT FROM SOFTWARE PROS

RETURN POLICY
Our "No Hassle" policy means if you're not satisfied with a product from Programmer's, simply return it within 30 days for a refund. Some manufacturer's products cannot be returned once disk seals are broken, so check before you buy.

INTERNATIONAL SERVICE
Take advantage of our International business number for details on export charges and exchange rates. Payments should be made in U.S. dollars.

DEALERS AND CORPORATE ACCOUNTS
Call and ask for our catalog and special volume discounts.

UNBEATABLE PRICES
We'll match lower nationally advertised prices.

In NY: **914-332-4548**
Customer Service: **914-332-0869**
International Orders: **914-332-4548**
Telex: **510-601-7602**
Fax: **914-332-4021**

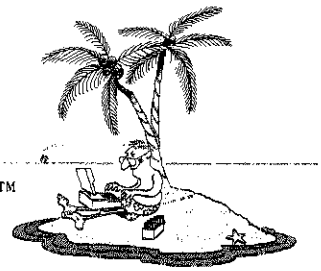
Call or Write for Latest Free Catalog!

1-800-445-7899

Programmer's Paradise™

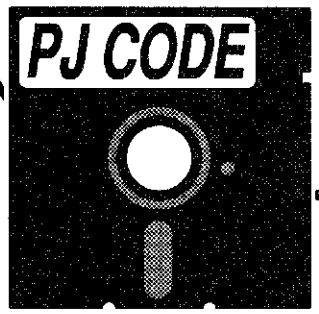
A Division of Magellan Software Corp
55 South Broadway, Tarrytown, NY 10591

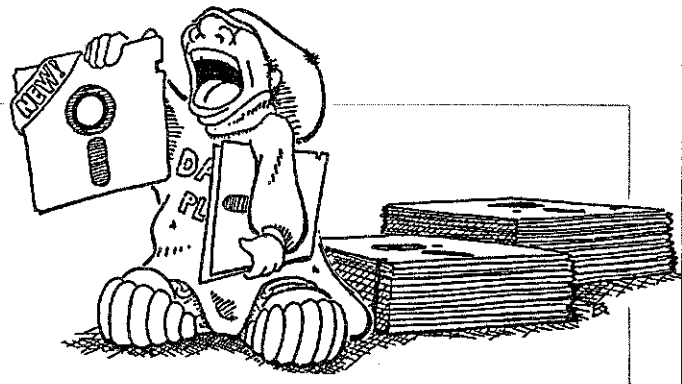
Number 200 on the Reader Service Card



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Git yer *PJ Code DiskScription!*





HOT CODE– COOL PRICES– WE DELIVER!

Typing in long code listings uses up a lot of valuable time, and calling up bulletin boards doesn't appeal to everyone. So what does the modern programmer do? Get a ***PJ Code DiskScription***, of course!

Now, for the first time ever, you can get a ***PJ Code DiskScription***--all the code listings from each and every issue of *Programmer's Journal* published in 1989, sent directly to you every other month for the **one time low charter subscription price of only $29.95.*** You'll get all the upcoming installments of Michael Abrash's EGA/VGA ASM code, more of Tom Swan's handy Pascal code, plus additional great code from the country's hottest programmer talent.

You'll receive the code disks automatically throughout the year. Learn from ***PJ*** code or use in it your own programs. Each disk contains all of the program listings published in ***PJ*** plus .EXE files, .OBJ files, benchmark source files and more!

If purchased separately, the regular price for these 6 disks is $60.00. **This special introductory offer saves you 50%, so order your *PJ DiskScription* today!**

HURRY! These charter subscriber prices will be honored only through **December 31, 1988**.

Programmer's Journal
Code DiskScription–
1989 price
~~$39.95~~

SPECIAL limited time
Charter Subscriber
offer only **$29.95***

* **Offer good only in US until 12-31-88.** Call or write for prices outside the United States. *PJ DiskScriptions* must be prepaid.

BE A CHARTER SUBCRIBER!

START YOUR *PJ Code DiskScription* TODAY!

PJ Code DiskScriptions must be prepaid. Return the card on the opposite page with your payment.
For faster service call toll-free: 800-234-0386 with your VISA or MASTER CARD.

*GUARANTEE: If you are dissatisfied with your **PJ Code DiskScription** for any reason, we will gladly send you a refund on all unmailed diskettes.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HARDCORE SOFTWARE

Stop fooling around. It's time to get hardcore about software. With Microsoft.

We'll give you all the resources you want. Tens of millions in R&D funding. Along with one of the most elementary tools for thinking — a door, which leads to your own private office. All backed by management that truly *does* speak your language, because they probably helped write it.

We're serious about software design. If you are too, then apply right now for one of these opportunities.

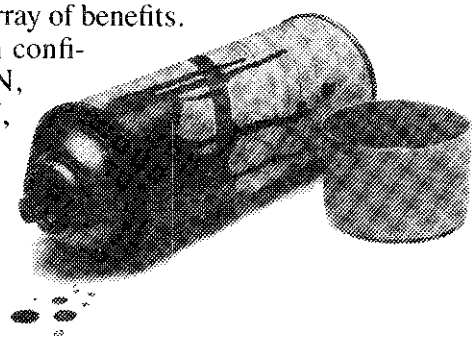
Software Design Engineers

We're working on everything from worldclass application software, compilers, operating systems, and networking to sophisticated graphics, powerful productivity software and more. You could be too, if you have programming experience and a background that includes 68000, Macintosh® Toolbox, Windows™, "C", micro's, 8086, UNIX™/XENIX®, or MS-DOS®.

There are opportunities to work with our teams in systems, applications, CD-ROM, languages, or networking.

If you qualify, Microsoft offers you an opportunity to live and work where the quality of life is high and the cost of living is low — the beautiful Pacific Northwest. Along with amenities such as a health club membership, workout facilities and parcourse, plus an array of benefits.

To apply, please send your resume in confidence to: MICROSOFT CORPORATION, Human Resources, Dept. CWSDE-1188PJ, 16011 N.E. 36th Way, Box 97017, Redmond, WA 98073-9717. We are an equal opportunity employer. No phone calls, please.



Microsoft®

Trademarks are registered to their respective companies.

Number 232 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

bination of key presses, so you may have eight interrupt handlers to handle mouse events. Function 24 requires AX = 24, CX = the trigger mask, and the ES:DX register pair pointing to the mouse interrupt handler.

Mouse function 25 lets you read the alternate mouse interrupt handler vectors and save that information so that you can switch back to them after using your own handlers. Function 25 requires AX = 25 and CX = the trigger mask for the interrupt handler you wish to read (only bits 5, 6, and 7 are valid in this case). Function 25 returns a –1 in AX if the alternate interrupt handler vector is undefined. If the alternate interrupt handler vector is defined, the alternate trigger mask is returned in CX and a pointer to the interrupt handler is returned in the BX:DX register pair (that's right, BX:DX—blame Microsoft).

Mouse function 26 alters the mouse's effective sensitivity, controlling the horizontal, vertical, and double speed sensitivities by controlling the ratio between the physical motion of the mouse and the number of motion units. As you increase the sensitivity, the number of motion units for a given displacement of the mouse will increase. Since the number of pixels that the mouse cursor moves is derived from the mouse driver motion units, the mouse will become more responsive as the sensitivity increases. Function 26 works in parallel with functions 15 and 19 to control total mouse responsiveness. Function 26 is not set to its default values of 50 when function 0 is called. Function 26 requires AX = 26, BX = horizontal sensitivity (0–100, 50 is default), CX = vertical sensitivity (0–100, 50 is default) and DX = double speed sensitivity (0–100, 50 is default).

Mouse function 27 allows you to read the mouse sensitivity. Function 27 requires AX = 27 and returns BX = horizontal mouse units/pixel (0–100), CX = vertical mouse units/pixel (0–100), and DX = speed threshold.

Mouse function 28 is an internal Microsoft function used by the Microsoft Inport mouse.

Mouse function 31 attempts to unhook the mouse driver by unhooking its interrupt 10h stub and the mouse hardware interrupt vector. Function 31 disables hardware interrupts for the mouse and returns the old (pre-mouse driver) interrupt 33h pointer to the caller so that the caller may restore the vector if he or she wishes. Function 31 requires AX = 31 and returns the pre-mouse driver interrupt vector for interrupt 33h in the ES:BX register pair (ES:BX—more Microsoft uniformity).

Mouse function 32 re-enables the mouse driver and hooks up interrupt 10h and the hardware interrupt vectors. Obviously, this will work only if you have not restored the interrupt 33h vector with its pre-mouse value. Function 32 requires only AX = 32.

Mouse function 33 is analogous to function 0. Function 33 requires AX = 33 and returns AX = –1 if the driver is installed and returns the button count in BX.

Mouse function 34 allows you to set the language byte. This is a byte value interrogated by certain Microsoft utilities to determine the language being used. This is one of those "what in the world is this doing here?" functions. Function 34 requires AX = 34 and BX = the language number. The languages are numbered 0 = English, 1 = French, 2 = Dutch, 3 = German, 4 = Swedish, 5 = Finnish, 6 = Spanish, 7 = Portuguese, and 8 = Italian.

Mouse Function 35 allows you to read the language byte. Function 35 requires AX = 35 and returns the language byte in BX.

Mouse function 36 returns information about the mouse driver software and hardware. Function 36 requires AX = 36 and returns two items. The mouse driver version is returned in BX—the high byte is the major revision level, and the low byte is the minor revision level. The mouse type is returned in the high byte of CX; 1 = bus, 2 = serial, 3 = inport, and 4 = auxport (PS/2). The interrupt request line used by the mouse is returned in the low byte of CX. For example, a serial mouse connected to COM1 and using Microsoft driver Version 6.11 will return 0611h in BX and 0204h in CX.

Michael Burton
Senior Software Engineer
Key Tronic Corporation
Spokane, WA

What have YOU been missing?

C Programmer's Toolbox Volumes I & II

Create better, faster, higher quality and easier to read programs in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

Easy to use. Unlimited program sizes. Online documentation...

CFlow™:	Determine program hierarchy, external R/T library functions, etc.;
CLint™:	More rigorously check program syntax;
CPrint™:	Beautify source programs to user selected formats;
CXref™:	Cross reference and check symbol usage;
CritPath™:	Determine a program's critical path;
PMon™:	Monitor program/OS execution; and more...

At $79.95 per volume or $130 for both with a 30 day money back guarantee, the Toolbox is simply the best value today. The Toolbox works with your existing C compiler(s) and enhances your development environment.

Call and Order Today
Visa, MasterCard Accepted

MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781
"The C Tool Specialists"

Circle number 94 on the Reader Service Card

More on Efficient Jump Tables

Here is one more approach to efficient jump tables (Tom Swan, "A Case of the Jumps," *PJ* 6.3; Adrian Crum and Tom Swan, "Letters," *PJ* 6.4). This tack answers some of Adrian Crum's complaints and has some of the nice features of Tom Swan's approach.

If a jump table is large and sparse, a two-level table can be used. For example, suppose ASCII codes are to be used to control jumping and only a few of the codes result in jumps to useful code while the rest are errors. We can set up a table of 128 bytes (call it CODETABLE) each of which contains an offset into another table (JTABLE) that contains the actual jump locations.

In Tom Swan's approach we'd jump to JTABLE[CODE] while in my approach we'd jump to JTABLE[CODETABLE[CODE]]. Thus we have an extra level of indirection, so we're not quite as efficient as Tom Swan, but for certain cases the storage used is much less. For example, if 4 bytes are needed for a pointer and there are 128 entries, Tom Swan's table would use 512 bytes. If only N codes really need to be distinguished, our table would use 128 + N * 4 bytes. So as long as N is less than 96, we save space. In fact, unless space is at an absolute premium, we would probably choose Tom Swan's approach for all but very small values of N.

Applying this technique to Adrian Crum's example (four ASCII codes cause us to dispatch, and all the rest are errors) requires 128

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

POP UP YOUR SYMBOLS

Get *FANSI* by adding ***hypertext capabilities*** to your C editing environment. Gain access to all your macro, function and typedef definitions instantly!

See CC-RIDER
SEE WHAT YOU CAN DO!

INCLUDES:

CCSYM Standard C "symbol compiler" which builds the symbol database used by CC-RIDER. Supports Microsoft C v5.1 extensions.

CCRIDER Pop-up program to recall your symbol definitions in a special window from any program.

USE YOUR OWN EDITOR

C.O.D., VISA, MASTER CHARGE and AMEX.

ONLY $89

Western Wares
Box C Norwood, CO 81423
(303) 327-4898

Number **100** on the Reader Service Card

TurboC, Lattice C, and QuickC

"c_wndw"In Windows, Menus & File Access

Now there is an affordable, complete set of functions to produce end-user DOS applications in the C language. "c_wndw" is a very complete toolkit, providing the following facilities

- o Color and cursor control
- o Pull-down and pop-up menus, and dialog lines
- o Field formatting and string handling
- o Entry editing of numbers and dates
- o Stacked, overlapping windows
- o File access error trapping and handling
- o Access to dBASE III plus '.DBF' files
- o Hashed random file access
- o Function key mapping and control
- o Undo and Help capability

For a license fee of $35, we provide a full set of object libraries for a C compiler, the full manual, and 19 sample programs. (S&H $2, GA res. add $1.85 tax). Specify your C compiler. Source code is available for $95.

To find out more about the "c_wndw" toolkit, write or call for information to Marietta Systems - (404) 565-1560
P.O. Box 71506, Marietta, GA 30007 Visa-M/C

Trademarks: "c_wndw", dBASE, QuickC, Turbo C, Lattice C

Number **223** on the Reader Service Card

PROTECT INTELLIGENCE WITH INTELLIGENCE

PROTECH KEYs:

- ❑ are totally transparent to the end user
- ❑ allow unlimited back up copies
- ❑ free up disk drives
- ❑ do not interfere with peripheral operations
- ❑ are easy to install
- ❑ protect software developers' revenues
- ❑ provide the highest level of protection available

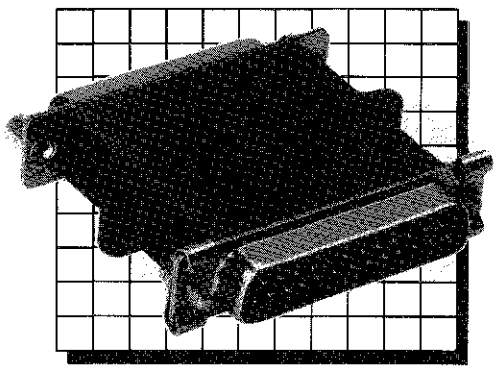
PROTECH'S KEYs provide the highest level of protection because of:

- ❑ assembler based routines (not drivers)
- ❑ encrypted routines
- ❑ physically unique hardware KEYs
- ❑ sophisticated software installation advice

THE KEY

The KEY provides effective software protection while insuring customer satisfaction. The KEY is a random response device which is designed for identically reproduced software packages.

SOFTWARE PROTECTION KEYs BY PROTECH



WE HAVE THE *KEY* TO SOFTWARE COPY PROTECTION

For a demonstration package or additional information, please write or call:

1-800-843-0413

THE MEMORY KEY

The MEMORY KEY is a programmable software protection device. Each byte of memory can be addressed in groups or individually. Possible applications for the MEMORY KEY include:

- ❑ modular package control
- ❑ serialization
- ❑ customization
- ❑ access control to PCs

Information stored in the MEMORY KEY can also be altered by your application software in the field. This dynamic writing to the MEMORY KEY opens up unlimited possibilities. Possible applications for the dynamic read/write function include:

- ❑ demo control
- ❑ software leasing
- ❑ updating modules in the field
- ❑ any "counter" operation

PROTECH MARKETING, INC.
1804 W. Southern Parkway • Building A-112
Durham, North Carolina 27707
(919) 490-4970 FAX (919) 490-4974

Number **228** on the Reader Service Card

PRODUCED 2014 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

bytes for the ASCII CODETABLE and 5 entries in the JTABLE. The fifth entry points to an error routine. Thus, 128 bytes (128 + 5 * 4) are required for the jump table.

As Tom Swan notes, Adrian Crum's search is just too slow. If a table is really as sparse as the one in Adrian Crum's example and no space can be spared for a jump table, we'd probably use a nested IF statement with the tests being done in order of highest probability of yielding a true result.

Michael Frame
Washington, DC

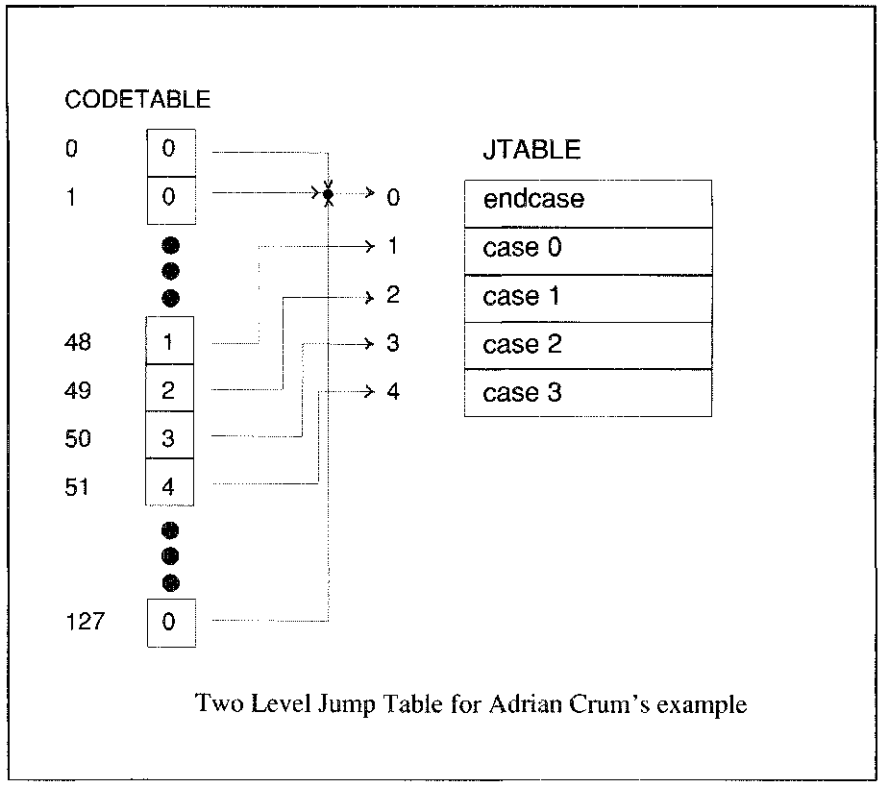
Tom Swan Concurs

This is a really good idea, and I agree with every point that Michael makes. I'd like to see a clever reader put this to good use and send *PJ* the Turbo Pascal code.

Tom Swan
Lititz, PA

Taking Exception

I must take exception to one or two points in Tom Swan's otherwise excellent article "Programming on the Variant Express" (*PJ* 6.5). He states, "A big plus in BASIC is the ability to dimension an array at runtime. Many people have *rightly* [emphasis mine] criticized Pascal for requiring array sizes to be hard-wired into the source text."



Two Level Jump Table for Adrian Crum's example

One has to recognize the genesis of both BASIC and Pascal. Both were invented as *training* languages. Neither was ever intended to be used as a serious *implementation* language. Within this context, there is little doubt that Pascal is far superior to BASIC, *primarily because of the restrictions* (such as that dealing with array declarations) *which it would place upon the user.*

Anyone who does not believe this should try teaching a course in program design to people who already know only BASIC as opposed to teaching a group who know only Pascal.

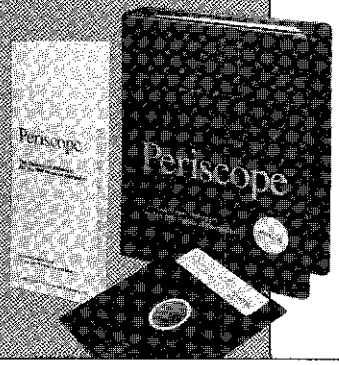
As an old ALGOLer myself, I personally miss the ability to declare array dimensions at runtime. I also miss the ability to declare "OWN" variables and the ability to use such statements as

```
A := if B = 0 then 1
       else 2;
```

but I ain't about to give up the convenience and power of the Borland product, or the wealth of support products available for it just because of such relatively minor inconveniences. (Obviously, Swan isn't either, since he goes to such lengths to invent such a powerful set of workarounds for the shortcomings of the language. I wish I'd thought of his technique for handling variant records myself!!)

Continued . . .

The New Periscope Version 4



...Gives you all the right stuff for debugging! No matter which model you pick, you have the same powerful software to help you track down hard-to-find bugs fast.

New in Periscope Version 4:

- View local symbols from Microsoft C
- Debug Microsoft Windows applications
- Set breakpoints in PLINK overlays
- Improved source-level support
- Monitor variables in a Watch window
- 80386 debug register support
- Debug using a dumb terminal
- PS/2 watchdog timer support
- Use mixed-case symbols

"The best keeps getting better... Once again, Periscope has raised the industry standard for debuggers!"

—David Nanian
BRIEF co-author

Periscope I includes a NEW full-length board with 512K of write-protected RAM; break-out switch; software and manual for **$695.**

Periscope II includes break-out switch; software and manual for **$175.**

Periscope II-X includes software and manual (no hardware) for **$145.**

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 10 MHz is **$1395.**

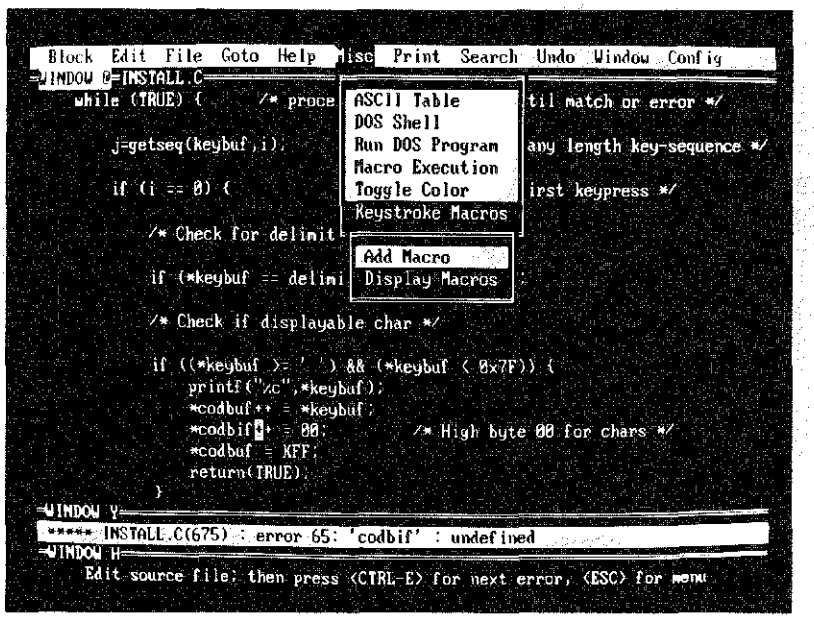
Call Toll-Free for free information or to order your Periscope today!

MAJOR CREDIT CARDS ACCEPTED. ***800-722-7006***

The Periscope Company, Inc.
1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404/875-8080

Circle number **6** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


Block Edit File Goto Help Misc Print Search Undo Window Config
WINDOW 0=INSTALL.C
while (TRUE) { /* proce til match or error */
j=getseq(keybuf,i); any length key-sequence */
if (i == 0) { irst keypress */
/* Check for delimit
if (*keybuf == delimi
/* Check if displayable char */
if ((*keybuf >= ' ') && (*keybuf < 0x7F)) {
printf("%c",*keybuf);
*codbuf++ = *keybuf;
codbif++ = 00; / High byte 00 for chars */
*codbuf = KFF;
return(TRUE);
}
WINDOW 9=
***** INSTALL.C(675) : error 65: 'codbif' : undefined
WINDOW 8=
Edit source file; then press <CTRL-E> for next error, <ESC> for menu
ASCII Table
DOS Shell
Run DOS Program
Macro Execution
Toggle Color
Keystroke Macros
Add Macro
Display Macros

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

NEW VERSION 3.0

- Best Multi-Level Undo
- Regular Expressions
- Pop-Up ASCII Table
- Pull-Down Menus
- Compiler Support
- Column Blocks

Until now, if you wanted the best Undo, the best compiler support, regular expressions and column blocks you chose BRIEF™. If you wanted unlimited keystroke macros, the best configurability, "off the cuff" command language macros and blazing speed, you chose VEDIT PLUS.®

Now the Choice is Easy

The all new VEDIT PLUS 3.0 gives you the best Undo of any editor, the best compiler support, unequaled windows, true regular expressions and extensive new features. We're leading the way with easy to use pull down menus, context sensitive help, a pop-up ASCII table, new printing options and much more. Incredibly, VEDIT PLUS 3.0 is now twice as fast as before and, at only 60K in size, it loads fast!

Completely Configurable

Change a few keys or redefine the entire keyboard. VEDIT PLUS adjusts to your editing style in minutes. You can even create new editing functions using simple keystroke macros or fine tune existing ones. VEDIT PLUS is so configurable that it easily emulates other editors and word processors (WordStar and Word Perfect emulation included). Quickly access editing functions with a single key or through the pull-down menus.

Try before You Buy

We challenge you to experience the dazzling performance and exceptional features that make VEDIT PLUS the best choice. Our evaluation disk includes the complete editor.* Learn VEDIT PLUS using our extensive "training" macro that gives instructions in one window while you experiment in another. See for yourself why no other macro language comes close.

Call for your free evaluation copy today. See why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Supports the IBM PC, XT, AT and PS/2 including DESQview, Microsoft Windows, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, CP/M-86 and FlexOS. (Yes! We support windows on CRT terminals.) $185.

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PC-MOS/386 is a trademark of The Software Link, Inc. CP/M-86 and FlexOS are trademarks of Digital Research. MS-DOS, OS/2 and XENIX are trademarks of Microsoft. DESQview is a trademark of Quarterdeck Office Systems.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, WYSE 700, Amdek 1280 and Others.
*Free evaluation disk is fully functional and can even edit small files.

Number 56 on the Reader Service Card

FREE EVALUATION COPY* Call 1-800-45-VEDIT

- Fully Network Compatible
- Call for XENIX and OS/2 versions
- 30 Day Money-back guarantee

Features of VEDIT PLUS 3.0

- Simultaneously edit up to 37 files of unlimited size.
- Variable sized windows; multiple windows per file.
- Execute DOS commands and other programs.
- Flexible "cut and paste" with 36 "scratch-pad" buffers.
- Block operations by line, character or column.
- Search with pattern matching or regular expressions.
- Configuration—determine your own keyboard layout, create your own editing functions, support any screen size.
- Select window colors, support 43 line EGA, 50 line VGA.

EASY TO USE

- Modern pull-down menu system. Pop-up ASCII table.
- Context sensitive on-line help is user changeable.
- Multi-level Undo (100 to 1000 levels). Undo keystroke by keystroke or line by line.
- On-line integer calculator (also algebraic expressions).
- Keystroke macros speed editing, menu function "hot keys."

FOR PROGRAMMERS

- Automatic Indent/Undent for "C," PL/I, PASCAL.
- Match/check nested parentheses, e.g. "{" and "}" for "C."
- Flexible macro runs popular compilers and automatically moves cursor to each error in your program. Easily changed to support new compilers and assemblers.

FOR WRITERS

- Word wrap, paragraph formatting and justification.
- Convert to/from Wordstar and mainframe files.
- Flexible printing; fully adjustable margins and Tab stops.

MACRO PROGRAMMING LANGUAGE

- If-then-else, looping, testing, string compare, branching, user prompts, keyboard input, 24 bit algebraic expressions.
- Flexible windowing—forms entry, select size, color, etc.
- Simplifies complex text processing, formatting, conversions.
- Extensive 400 page manual with hundreds of examples.

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299 • Telex 701821 • Fax (313) 996-1308

CompuView

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOS & OS/2
Free Demo Diskette Available

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! *If you use* VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
  Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
  Analyzes an entire collection of source files with a single statement.
- **Nested Include Files**
  Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
  Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
  Shows the entire include file hierarchy for each source file analyzed, and all of the parent source files for each include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! *If you use* VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
  Handles C, assembler, or both.
- **Flexible**
  File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
  Easily handles systems containing 100 source files or more.

VersiMAKE™	$125
VersiCREF™	$75
Both	$150

(Outside U.S., add $5 for shipping and handling)

800-334-4096
(In NJ, 609-871-0202)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.
73 East Lane, Willingboro, NJ 08046

Circle number **158** on the Reader Service Card

I feel very strongly, however, that the criticism of Pascal for so aptly meeting its design goal, and the elevation of BASIC for failing to meet the same goal (although, to be fair, the desirability of strong typing was unrecognized when BASIC was invented) is unwarranted and lends aid and comfort where it doesn't belong. Perhaps I'm being iconoclastic. I hope so.

W.G. Madison
Greenbelt, MD

Tom Swan Apologizes

Maybe I was a little hard on Pascal, poor ol' girl. It's not her fault she was born with non-dimensionable arrays, wishy-washy strings, and archaic I/O. I guess it's true: We do always hurt the ones we love. Shame on me! I should watch my wicked tongue.

Tom Swan

Undocumented DOS Functions

This is a response to M. Steven Baker's column on undocumented functions, etc., in Volume 6.4. This sort of thing has bothered the hell out of me from the very start. I have spent hours, and days, even, documenting for myself various DOS secrets.

Here, then, are a couple undocumented features of DOS and of the 80x86 family of CPUs. I'll start with the undocumented CPU instructions I've stumbled across. The first of these jumped up and startled me when I switched from MASM 4.0 to 5.0. I reassembled my editor (some 360 KB of source, but I comment liberally) as a test of speed. The resulting ED.COM file was some 16 bytes smaller than the version assembled with MASM 4.0. At first I thought, ah ha! Microsoft has finally solved the NOP problem after a short jump, and I missed a couple of those. NIX! That was not the case. Take a look at the following dumps:

```
>>>AFD print out     10-7-1987     7:15

;THIS IS A DISASSEMBLY OF THE ROUTINE TOGGLE_CURSOR FROM ED2_BIOS.
;THIS FILE WAS ASSEMBLED WITH MASM 5.0.

54C7:15EA  FF068406       INC     W/[0684]
54C7:15EE  83             DB      83  ;what is this doing here?
54C7:15EF  26840603BB     TEST    ES:[BB03],AL ;TEST???????
54C7:15F4  8E06813E       MOV     ES,[3E81] ;crazy, man!
54C7:15F8  7206           JC      1600
54C7:15FA  00B87403       ADD     [BX+SI+0374],BH
54C7:15FE  BB8606         MOV     BX,0686
54C7:1601  031E8406       ADD     BX,[0684]
54C7:1605  031E8406       ADD     BX,[0684]
54C7:1609  8B07           MOV     AX,[BX]
54C7:160B  A30001         MOV     [0100],AX
54C7:160E  A3CB15         MOV     [15CB],AX
54C7:1611  C3             RET

;HERE IS THE DISASSEMBLY FROM THE SAME FILE ASSEMBLED WITH MASM 4.0

>>> AFD  print out  10-7-1987     7:49

64C7:15EA  FF068406         INC     W/[0684]
64C7:15EE  812684060300     AND     [0684],0003
64C7:15F4  BB8E06           MOV     BX,068E
64C7:15F7  813E720600B8     CMP     [0672],B800
64C7:15FD  7403             JZ      1602
64C7:15FF  BB8606           MOV     BX,0686
64C7:1602  031E8406         ADD     BX,[0684]
64C7:1606  031E8406         ADD     BX,[0684]
64C7:160A  8B07             MOV     AX,[BX]
64C7:160C  A30001           MOV     [0100],AX
64C7:160F  A3CB15           MOV     [15CB],AX
64C7:1612  C3               RET

;HERE IS THE SOURCE.

TOGGLE_CURSOR proc near
;This routine will allow the user to change the shape of the
;overtype cursor. Each call to the routine changes the shape,
;moving through a circular list of 4 possibilities. There is one
;set of 4 for the color card and one set of 4 for the monochrome
;card. The choices are:
;DEFAULT,DOS_CURSOR,FULL_BOX,DOUBLE_DOS_CURSOR
;----------------------
       inc DS:CURRENT_CURSOR
       and DS:CURRENT_CURSOR,011b ;blend out all but bottom 2 bits
       mov bx,offset DS:COLOR_CURSORS ;assume trouble is on color card
       cmp DS:SCREEN_SEGMENT_POINTER,COLOR_BUFFER
       je TOG_CUR_1

       mov bx,offset DS:MONO_CURSORS ;correct table pointer
TOG_CUR_1:
       add bx,DS:CURRENT_CURSOR
       add bx,DS:CURRENT_CURSOR ;point into table to new cursor type
       mov ax,DS:[bx]
       mov DS:DEFAULT_CURSOR,ax
       mov word ptr DS:CURTYP[1],ax
       ret
TOGGLE_CURSOR endp
```

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

UNLIMITED MEMORY C++

By now, you've probably noticed that the DOS C++ translator or compiler you bought suffers from one or more fatal drawbacks:

- "Cfront: out of memory" AARRRGH!!!
- 8088 code generation only
- Limited range of C compiler support

Intek offers you more! *For just $495.00 you get:*

- AT&T C++ translator v1.2 running in 80386 protected mode extended MS-DOS, allowing 4 GB access
- Support for a broad range of C compilers, including METAWARE, MICROSOFT, TURBO, WATCOM; call about your unique requirements
- Ask about our 16 MB 80286 version!

INTEK
Integration Technologies, Inc.
1400 112th Ave. S. E., Suite 202
Bellevue WA 98004
(206) 455-9935

Requires 80386, 640K (1 MB or more recommended), C compiler. Note that 4 GB access refers to compile time; runtime access depends upon your C compiler. Metaware, Microsoft, and Watcom are all trademarks of their respective companies; Turbo C is a trademark of Borland Intl; MS-DOS is a trademark of Microsoft. Intek C++ is licensed from AT&T.

Number **239** on the Reader Service Card

Personal REXX: The batch alternative. Now.

Personal REXX is a powerful general-purpose language bringing mainframe command and macro language capabilities to the PC.

Personal REXX offers you:

- A superset of **IBM CMS REXX.**
- The ability to develop **powerful batch procedures.**
- **Compatibility** with **IBM SAA REXX** command language.
- **Windowing** and **menu support.**
- **Easy prototyping.**
- The ability to **interface with KEDIT** text editor, and other applications.

Personal REXX runs on the IBM PC, AT, PS/2, and true compatibles using DOS 2.0–DOS 3.3. With Personal REXX, you can enjoy the many benefits of the macro language of the future. Today.

MANSFIELD
Software Group
P.O. Box 532
Storrs, CT 06268
(203) 429-8402

Personal REXX Version 1.60 available at $125 plus $3 shipping. MC, VISA, American Express.

KEDIT is a trademark of the Mansfield Software Group, Inc.; IBM, IBM PC, AT, and PS/2 are trademarks of International Business Machines.

Number **233** on the Reader Service Card

TRUE MULTITASKING

With

MultiDos Plus

"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: $24.95 OR $99.95 with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

Number **109** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Eco-C88 C Compiler with Cmore Debugger

Professionals prefer the Eco-C88 C compiler for ease of use and its powerful debugging features. Our "picky flag" gives you nine levels of lint-like error checking and makes debugging easy.

"I'm very impressed with the compiler, editor, and debugger. I've tried quite a few different compilers for the PC and have given up on all of the others in favor of yours... I've gotten to the point where I download C code from a DEC VAX/VMS system just to be able to compile it with the picky flag set at 9. It finds lots of things VMS totally ignores..."
JS, Oak Ridge, TN

The Eco-C88 compiler includes:

- A full-featured C compiler with 4 memory models (up to 1 meg of code and data) plus most ANSI enhancements.
- Without a doubt, the best error checking you can get. We catch bugs the others miss, making you much more productive.
- Cmore is a full-featured source code debugger, not some stripped-down version.
- Robust standard library with over 230 useful (no "fluff") functions, many of which are System V and ANSI compatible. Full source is available for only $25.00 at time of order.
- CED, a fast, full screen, multiple-window program editor with on-line function help. You can compile, edit, and link from within CED.
- cc and mini-make utilities included that simplifies the most complex compiles.
- Users manual with over 150 program examples (not fragments) to illustrate how to use the library functions.
- Fast compiles producing fast code.

Our Guarantee: Try the Eco-C88 compiler for $99.95. Use it for 30 days and if you are not completely satisfied, simply return it for a full refund. We are confident that once you've tried Eco-C88, you'll never use anything else. Call or write today!

Orders: **1-800-952-0472**
Info: **1-317-255-6476**

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220

ECOSOFT

Circle number 8 on the Reader Service Card

The problem occurs at the second instruction in the routine. MASM 4.0 assembles this instruction as it should. MASM 5.0, on the other hand, seems to assemble complete trash. *Yet the routine runs like a charm!* A bit of scrutiny into SYMDEB yields the following dump:

```
Microsoft (R) Symbolic Debug Utility  Version 4.00
Copyright (C) Microsoft Corp 1984, 1985.  All rights reserved.

Processor is [8086]
-N ED.COM  ;THIS IS A MASM 5.0 FILE
-L
-U 15D8
4F87:15D8 FF068406        INC     Word Ptr [0684]
4F87:15DC 8326840603      AND     Word Ptr [0684],+03
4F87:15E1 BB8E06          MOV     BX,068E
4F87:15E4 813E720600B8    CMP     Word Ptr [0672],B800
4F87:15EA 7403            JZ      15EF
4F87:15EC BB8606          MOV     BX,0686
4F87:15EF 031E8406        ADD     BX,[0684]
4F87:15F3 031E8406        ADD     BX,[0684]

-N B:ED.COM  ;THIS IS THE OLD MASM 4.0 FILE
-L
-U 15D8
4F87:15D8 FF068406        INC     Word Ptr [0684]
4F87:15DC 812684060300    AND     Word Ptr [0684],0003
4F87:15E2 BB8E06          MOV     BX,068E
4F87:15E5 813E720600B8    CMP     Word Ptr [0672],B800
4F87:15EB 7403            JZ      15F0
4F87:15ED BB8606          MOV     BX,0686
4F87:15F0 031E8406        ADD     BX,[0684]
4F87:15F4 031E8406        ADD     BX,[0684]
-
```

Notice that SYMDEB shows the second instruction (AND word ptr [xxxx],+03) as being what was intended. *However*, the Intel documentation on the 80x86 family and Russell and Alexey's Blue Bible say that opcodes beginning with 83H are not used. This remains true for the 80386 at least as far as the Intel *Programmer's Reference Manual* is concerned. Clearly, Microsoft knew about this instruction, and Putkammer (the guy who wrote the AFD debugger) did not. This sort of thing can be carried to extremes and drive the poor programmer straight up the wall. A whole bunch of the logical instructions of the form,

```
logical_instruction word ptr,
  immediate byte
```

behave in the above manner. I have not documented all of them, nor am I sure how they work. Is the byte sign extended or not? Nor is there a way, short of coding the instruction directly, of persuading MASM 5.0 to use the old codes.

Another undocumented CPU feature concerns the AAM instruction. The opcode for this instruction is D4 0A. The instruction divides AL by 0A = 10d and stores the quotient in AH and the remainder in AL. The instruction then sets the flags according to some scheme or other. After my experience with the AND instruction above, I was suspicious of the 0A in the opcode. Yes, you guessed it! The opcode D4 0# will divide AL by # and store the result in AH and the remainder in AL, setting the flags according to the same scheme as AAM's—a cute trick for formatting packed integers mod #.

My last goody concerns INT 2FH (the Multiplex Interrupt) and the exchange of letters concerning it in recent issues. I, too, have used INT 2FH for TSRs for some time. Recently, however, while writing a TROFF program, I noted that IBM says you can use this interrupt to communicate with the resident portion of PRINT. I thought why not build a spool feature into TROFF as follows: If the spool flag is entered, write the file to disk, use INT 2FH function AL = 1 (submit file) to tell PRINT about the file, and let PRINT deal with the file in the background. This works just fine (now!) except for the fact that this call to PRINT in DOS 3.10 will trash the registers SI and DI. This is really grand when the program is written in C. It took me four hours to find this bug. Of course, the documentation says nothing about these registers being destroyed. I suppose we should be grateful that the segment registers are at least preserved.

Larry Smith
D 3401 Friedland 5
West Germany

Not Quite GOTOless

A short note by W.G. Madison in the September/October *Programmer's Journal* (p. 56) had us chuckling at the idea that any modern programmer would ever use a GOTO statement. But at the risk of being convicted of Structured Programming heresy, I report that I do use GOTOs regularly, in otherwise tightly structured and rigidly formatted Pascal code.

While I am strongly in favor of structured code, I am *more* in favor of efficient and understandable code. Unfortunately, I find that the dogmatic application of structured principles to every nuance of a program may be counter-productive. Structuring, by itself, is no panacea, and a rigid practice of structuring may actually produce harder-to-understand and less efficient code.

Code efficiency is not a problem if you believe that your compiler can transform *any* equivalent program into *exactly the same*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

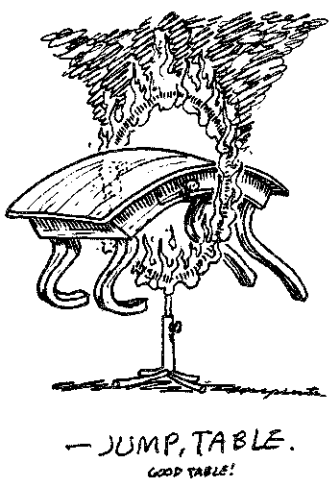
minimum code that the program requires. Unfortunately, such compilers do not exist, and will *never* exist, except as a goal (and I am not sure that I would like to debug the result from such a compiler anyway). It does not seem too much to expect that a programmer will guide the compiler into an efficient expression of the required code. And I am aware of no Computer "Science" or Structured Programming theorem that asserts that a fully structured program must also be the most efficient program, or, for that matter, the most easily understood program.

Code clarity has been seen by a number of authors to be the inverse of code complexity, and we do have a number of (controversial) measurements of code complexity. But I find it more reasonable to see a computer program as a *form of literature*, which, in its highest and best expression, communicates well to both computer and human "readers." As with other forms of literature, an individual opinion on the work often depends as much on the context of the reader as on the work itself. Even the same person may, at different times, have a different opinion of the same program code.

In the unending pursuit of clear code, I generally try to decompose a complex problem into separate procedures, any one of which can be separately understood and independently tested. I give meaningful names to variables and provide several *levels* of comments (although any particular comment will be at the same time too trivial for some readers and too detailed for others). I also work to reduce the number of variables in the code, and I try to eliminate state variables used solely for control purposes. Experience has taught me that such variables tend to make the program more difficult to understand. The flow-of-control of a program with a state variable depends upon the state value, and the state value, after some execution, can be deduced only by having mentally executed the previous steps of a flow-of-control that is not yet fully understood. Thus, the addition of state variables, simply to allow a given program logic to be expressed in a structured form, is usually counter-productive with respect of clarity. It may also have a small negative effect on efficiency. But such state variables (often Boolean "flags") are *necessary* to the expression of some program logic in a structured form.

While the general thrust of Structured Programming is an unarguable asset, dogmatic application, such as the unquestioning elimination of all GOTOs, probably should be challenged. In fact, that is now in progress. W.G. Madison's note (which prompted this reply) referenced Dijkstra's letter "GoTo Statement Considered Harmful" in the March 1968, issue of *Communications of the ACM*. More recently, the March 1987, issue of the same journal carried a letter by Frank Rubin under the title "'GOTO Considered Harmful' Considered Harmful." Extraordinary response followed, especially in the May, June, July, August, November, and December (1987) issues. I hope the result of the controversy will be a more scientific inquiry into the extent to which Structured Programming, by itself, satisfies *all* our programming goals and whether some limited discretion should be used in its application. I would like to see additional forms of control structure brought into the structuring system to avoid some of its worst aspects. But until this is done, I assume the responsibility to write the clearest code I can, and this sometimes means adding GOTOs to otherwise structured code.

Terry Ritter, P.E.
Blue Jean Software
Austin, TX ♦



OMNIVIEW
ROCK SOLID MULTITASKING!

Packed with the power you need, OMNIVIEW takes you beyond the limits of DOS.

- Time-slices multiple jobs
- Takes full advantage of EMS 4.0
- Optional support for '386 Systems
- Manages megabytes of programs
- Controls resident utilities
- Lets you cut and paste between programs

OMNIVIEW unleashes your PC & converts it into an easy-to-use, multitasking workstation. Whether you have an 80386 or 8088 based computer, OMNIVIEW will squeeze out every ounce of power.

Solid & bulletproof, it's 47K assembly language engine gives you all the power without added overhead or complexity. Just ask the reviewers.*

"Sunny Hill provides true multitasking for most PC applications. It performs remarkably well, causing almost no noticeable reduction in the performance of each program . . . The user manual is well organzied and easy to read . . . It is an excellent product."
PC Tech Journal Sept '87

"It does an amazing job of handling not only multitasking, but in many cases, concurrent processing as well. I was amazed at it's success in handling the "impossible" combinations I threw at it."
"It is one of the best designed & implemented programs I've ever worked with. It is also one of the most bulletproof."
"What sets it apart from the pack is the fact that it runs in a network environment."
Data Based Advisor Aug '87

OMNIVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:
(800) 367-0651
or send $89.95 + 3.00 S&H ($8.00 Intl.) to:

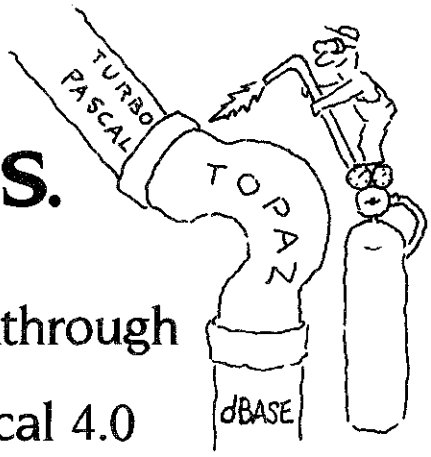
Sunny Hill Software
PO Box 55278, Seattle, WA 98155-5278

For more information call (206) 367-0650 *Omniview formerly Taskview
DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

Circle number 85 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

YOU'LL LOVE THESE UTILITIES.



SAYWHAT?! The lightning-fast screen generator

It doesn't matter which language you program in. With Saywhat, you can build beautiful, elaborate, colorful screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can all be displayed with as little as one line of code in *any* language. Batch files, too.

With Saywhat, what you see is *exactly* what you get. And response time is snappy and crisp, the way you like it. That means screens pop up instantly, whenever and wherever you want them.

Whether you're a novice programmer longing for simplicity, or a seasoned professional searching for higher productivity, you owe it to yourself to check out Saywhat. For starters, it will let you build your own elegant, moving-bar menus into any screen. (They work like magic in any application, with just one line of code!) You can also combine your screens into extremely powerful screen libraries. And Saywhat's remarkable VIDPOP utility gives all languages running under PC/MS-DOS, a whole new set of flexible screen handling commands. Languages like dBASE, Pascal, BASIC, C, Modula-2, FORTRAN, and COBOL. Saywhat works with all the dBASE compilers, too!

With Saywhat we also include a bunch of terrific utilities, sample screens, sample programs, and outstanding technical support, all at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed). **$49.95**

WE GUARANTEE IT!

IRON CLAD MONEY-BACK GUARANTEE.
If you aren't completely delighted with Saywhat or Topaz, return them within 30 days for a prompt, friendly refund.

TOPAZ. The breakthrough toolkit for Turbo Pascal 4.0

If you'd like to combine the raw power and speed of Turbo Pascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. You see, Topaz (our brand new collection of units for Turbo Pascal 4.0) was specially created to let you enjoy the best of *both* worlds. The result? You can create truly dazzling applications in a very short time. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you create outstanding, polished programs, fast. Think of it. With Topaz you can write Pascal code using SAYs and GETs, PICTURE and RANGE clauses, then SELECT and USE databases (real dBASE databases!), SKIP through records, APPEND data, and lots more.

In fact, we've emulated nearly one hundred *actual* dBASE *commands and functions*, and even added *new* commands and functions to enhance the dBASE syntax! All you have to do is declare Topaz's units in your source code and you're up and running!

The bottom line? Topaz makes writing sophisticated Pascal applications a snap. Data entry and data base applications come together with a minimum of code and they'll always be easy to read and maintain.

Topaz comes with a free code generator that automatically writes all the Pascal code you need to maintain a dBASE file with full-screen editing. Plus outstanding technical support, at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed). **$49.95**

ORDER NOW. YOU RISK NOTHING. Thousands of satisfied users have already ordered from us. Why not call toll-free, right now and put Saywhat and Topaz to the test yourself? They're fully guaranteed. You don't risk a penny.

SPECIAL LIMITED-TIME OFFER! Buy Saywhat?! and Topaz together for just $85 (plus $5 shipping & handling). That's a savings of almost $15.

To order: Call toll-free

800-468-9273

In California: **800-231-7849**
International: 415-571-5019

The Research Group
88 South Linden Ave.
South San Francisco, CA 94080

☑ **YES.** I want to try:

Saywhat?! your lightning-fast screen generator, so send _____ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee.

Topaz, your programmer's toolkit for Turbo Pascal 4.0, so send _____ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee.

☐ **YES.** I want to take advantage of your special offer! Send me _____ copies of both Saywhat?! and Topaz at $85 per pair (plus $5 shipping & handling). That's a savings of almost $15.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

☐ Check enclosed ☐ Ship C.O.D. ☐ Credit card

_____ Exp. date _____ Signature _____

THE RESEARCH GROUP

Number 198 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOS 4.0 Leaps into the OS Mix

by
Bill Madison

I was in the middle of a letter to the editor of *PJ* in which I deplored the hype surrounding OS/2 when the announcement of DOS 4.0 popped up from nowhere (well, actually, from CompuServe). I took a cold shower and started to read.

The thesis of my diatribe had been that a tool like OS/2, while perhaps admirably suited for certain classes of application mixes, is grossly incapable of providing a generally optimal operational environment. My arguments were based on ergonomics and linguistics—and were totally persuasive and guaranteed to cause hypertension. The DOS 4.0 announcement thus came like a breath of springtime.

Of course, the announcement was not a total surprise. The word on the street for some time had been that "DOS 3.4" was in Beta test. Equally "of course," no one could find out much about it. I was not even really sure whether it was to be MS or PC DOS. Rumors of a European multitasking "MS/DOS 4.0" still float around; but that's not this one.

Bear in mind in what follows that I have never actually used OS/2. "All I know is what I read in the papers."

Just What's Going on?

With that *caveat*, let's consider what appears to be the current situation and try to assess the future.

As a start, let's look at the announcement's list of the major features of PC DOS 4.0:

- Upward compatibility with DOS 3.3. DOS 3.3 will continue to be marketed.
- 32-bit disk (DASD) addressing; no more 32MB disk partition size limit. [What will this do for CD-ROM?]
- Choice of (a) command-line, (b) menu oriented, or (c) icon (i.e., graphics) oriented user interface, selectable at installation time.
- Built-in support for EMS 4.0 expanded memory, with the ability to use extended memory as expanded memory.
- Certain DOS functions (BUFFERS, FASTOPEN, VDISK) make use of EMS memory if it is present. This will free up for other applications the conventional memory these functions now occupy.
- A full screen installation utility. [Of course, the need for such a utility implies that DOS 4.0 will not be a "drop-in" as earlier versions were. Ah! The price of progress!]
- Several additional support functions for improved EGA/VGA text-mode operation and for improved display adapter/printer graphics support.
- Other changes intended to solve many of the minor (to me) annoyances traditionally imposed upon us poor developers. (Examples: BACKUP now accepts an unformatted target diskette [but no indication of whether it's any faster]; DEBUG provides EMS support; DEL/ERASE optionally requires verification prior to deletion; . . .)

Questions now arise: What does this all mean as far as Big Blue's future marketing intentions are concerned? Is there any truth in the rumors that IBM's confidence in OS/2 has been seriously eroded? In My Humble Opinion, several things are apparent.

A power higher than Armonk appears to have exerted itself.

Despite its relatively robust implementation, its flexibility, and its ease of use, DOS has had several recognized deficiencies. (I do not include the lack of multitasking or multiuser support among them, for the moment.) Many of these deficiencies have apparently now been addressed within the context of DOS 4.0. [*Big* sigh of relief. The DOS we have known and loved for so long is not the faded harridan we had thought. She just needed a face lift and a fanny tuck!]

OS/2, while it has addressed many of these same problems, has done so within the context of a completely new set of capabilities and features that many (most?) installations neither needed nor wanted. Inclusion of these features represents a waste of fiscal and operational resources.

IBM/Microsoft, adhering to their traditional policy of "going where the money is," have elected to take a giant step "backwards," incorporating some of the more eagerly awaited features of OS/2 into DOS and ignoring (within the DOS context) the remainder. This rather restores my faith in the power of the marketplace and the free enterprise system.

In providing for three different user interfaces to DOS (command line, text windows, graphics windows), IBM/Microsoft have set an important precedent. We can only hope that this presages more modular microcomputer operating systems. One day (DOS 4.5? 5.0?) the user may be able to install only those families of features that are needed, ignoring (and thus not expending the hardware resources needed to support) those not desired. Of course, you might validly argue that this is exactly what has been happening with the large number of TSRs available. If you want a file management shell, you install SCOUT or QFILER. If you want an enriched batch capability, you install EBL, and so on, *ad nauseam*.

A Multiuser/Multitasking DOS?

You could argue that DOS should be expanded to support a multiuser/multitasking environment. I think this argument is invalid for several reasons. First, were multiusing/multitasking or either to be incorporated, we would be in many respects right back to a mini-OS/2 unless the modular approach mentioned above were taken.

Second, direct EMS support, especially within the context of a choice of user interfaces, means that the hooks permitting such facilities to be developed by third party vendors are present; we won't be without such capabilities for long if they are really needed. And such facilities would have the desired modularity; either install them or don't—your choice.

Finally, if the rumored European DOS 4.0 does in fact exist, and if it is a multitasking DOS, how long can it be before the two are brought together into a single package?

[In my parochial way, I could argue, however, for better (or at least, better documented) TSR support. But then, I haven't seen the new documentation; maybe it's already there. How's that for *naïveté*?]

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FINALLY!

See us at booth R8700
COMDEX/Fall '88
November 14—18, 1988
Riviera Hotel, Las Vegas, Nevada

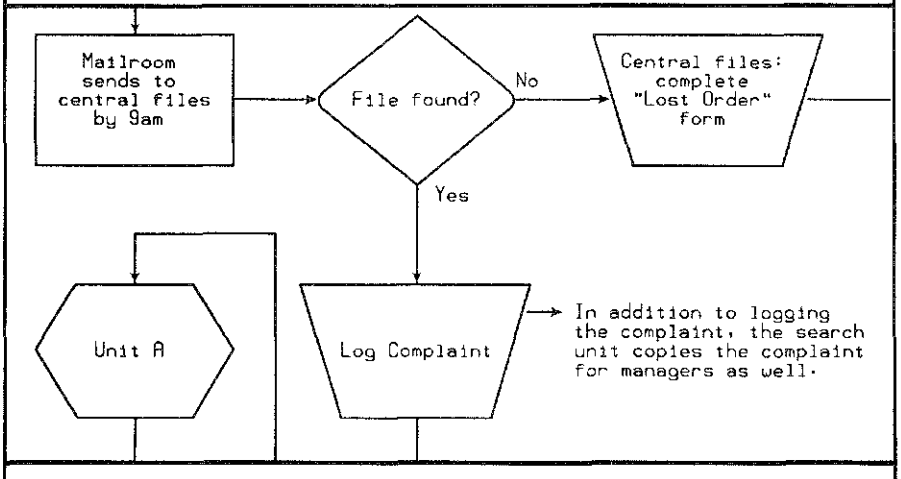
EASYFLOW

An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work. EASYFLOW is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With it you can quickly compose charts. More important, you can easily modify charts so they are always up to date.

- **Automatic:** Fully automatic text centering within shapes, both horizontally and vertically. Fully automatic line routing & re-routing.
- **Fast:** Written in assembly language for speed.
- **Large:** Charts up to 417 columns wide by 225 lines high. Chart too large for your printer? *EASYFLOW* automatically breaks the chart up & prints it in page size pieces.
- **Standard:** All standard flowcharting shapes included. Custom shapes can be ordered.
- **User friendly:** Don't take our word for it. PC Magazine says "*EASYFLOW lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts*". [March 10th 1987 issue, page 278.]
- **Mouse:** Optional but fully supported.
- **It prints:** On most popular matrix printers including IBM, Epson, Toshiba, HP LaserJet, LaserJet-Plus and many others.
- **It plots:** On HP7440, 7475, 7550, 7570, 7585B and compatible plotters.
- **It works:** We are contractually prevented from mentioning the name of the "big eight" accounting firm that purchased a world-wide site license, but we can tell you that they spent months evaluating all available flowcharting packages before choosing *EASYFLOW*.
- **Rush delivery:** Order by noon today (eastern time) and you'll have it tomorrow. Rush delivery charge is $10.50 (instead of $2.00) and is available only to continental USA and Canada.
- **Documented:** 100 page manual plus over 150 screens of context sensitive help.

EASYFLOW works on IBM PC's, IBM PS/2 and compatibles. Requires 384 K memory, DOS 2.0 or higher and an IBM CGA/EGA/VGA or Hercules monochrome compatible adapter card.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., cheque, VISA, Mastercard or Company PO.


Mailroom sends to central files by 9am
File found?
No
Central files: complete "Lost Order" form
Yes
Unit A
Log Complaint
In addition to logging the complaint, the search unit copies the complaint for managers as well.

The chart fragment above was produced on an HP LaserJet-Plus and is actual size and unretouched. Publication quality charts like this can be produced using only minutes of preparation time and seconds of print time.

HavenTree Software Limited
PO Box 1093-V
Thousand Island Park, NY 13692

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext 31
Telefax(G3): (613) 544-9632

Circle number **154** on the Reader Service Card

Conclusion

The DOS 4.0 announcement suggests that we may reasonably expect a continued evolution of DOS, and that we can expect that evolution to continue in directions that will provide "the greatest good for the greatest number."

Like it or not, IBM still has the power to set standards within the microcomputing world. Fortunately, a power higher than Armonk, namely the power of the marketplace, appears to have exerted itself.

If history repeats, as it so often does, we can expect IBM pronouncements that put a very positive face on the DOS 4.0 announcement. They will imply that the simultaneous support of two (let's face it) competing operating systems is:

(a) no big deal;

(b) according to our long range corporate plan;

(c) in the interest of our large base of single user customers, whom we would *never* abandon; or

(d) all of the above.

Choose one.

Hogwash. They screwed up. They misread the marketplace. But for once, the entire community of microcomputer users must ultimately benefit from the cross-fertilization that is apparently already taking place.

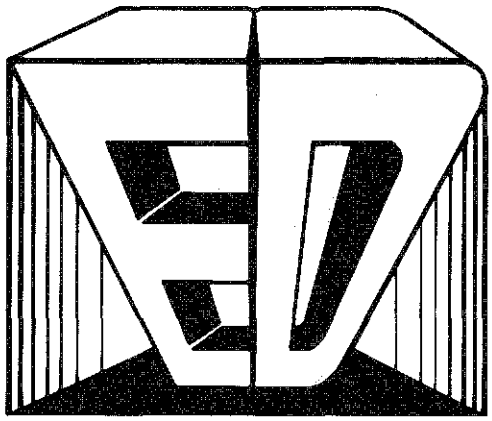
It would be foolhardy to expect that OS/2 will go the way of the PCjr, nor should it. Those installations having a *true* need for the facilities provided by OS/2 have (or will have) an operating system capable of filling their needs; those who do not have such a need will not be forced to pay the price of having major unused capabilities on their hands.

An increased burden will be placed on developers by the requirement to support their products under two different operating systems. If, however, this burden cannot be justified in the market place, it will not be assumed or it will be abandoned. This is as it should be.

Maybe some day we will be given an operating system with a sufficiently high degree of modularity (as suggested above) so that the DOS and OS/2 evolutionary paths can converge. Until then, *vive la difference!* ♦

Bill Madison is an independent computer consultant whose clients include Amtrak, the National Institutes of Health, the National Oceanic and Atmospheric Administration, the National Weather Service, the U.S. Department of Health and Human Services, and R.J. Reynolds Tobacco Co. Bill has been a member for nine years, chairman for seven years, of ANSI/X3/SPARC, the staff planning committee for the group within ANSI responsible for the development of industry standards relating to information systems. Readers may contact Bill at W.G. Madison and Associates, Ltd., P.O. Box 898, Greenbelt, MD 20770, (301) 552-7234 (CompuServe 73240,342).

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



TM

EDITING SYSTEM

ED is an editor that makes the very conception of other editors obsolete.

ED is an object-oriented open architecture system for editing programs and manuscripts, manipulating data files and creating snazzy demos. ED's commands respond with no perceptable delays. Programmers have complete simultaneous control over multiple source files. Functions for creating manuscripts are part of ED's design, with stream blocks, word-wrap, tabs, intelligent paragraph formatting, block justification, and pagination that behave correctly and execute instantly. ED can be used to create sophisticated sorts and filters. Macros, search and replace, column blocks, and block sorts allow records to be sorted and selectively removed, and fields to be added, rearranged, and removed.

Flexibility, extensibility, and programmability are realized by providing programmers access to the objects which ED itself manipulates. All aspects of ED's appearance and operation can be controlled by the programmer. ED, a restricted or enhanced form of ED, or any of ED's objects, can be embedded in the programmer's applications with no royalties. Objects such as dynamic arrays, windows, data entry windows, macros, menus, browsers, popup directories, and regular expressions are manipulated through normal C function calls.

ED • NUMBER OF BUFFERS LIMITED ONLY BY AVAILABLE RAM • NUMBER OF WINDOWS LIMITED ONLY BY SCREEN SIZE • **POPUP DIRECTORY FACILITY** • VIEW FILESPEC/SUBDIRECTORIES • FIND FILE ON DISK • SEARCH DISK FILES FOR TEXT • COPY • RENAME • DELETE • BROWSE • FILE STATUS • EDIT • EXECUTE • SORT • **SEARCH AND REPLACE** • FORWARD/BACKWARD • CASE SENSITIVE/INSENSITIVE • FULLY PARENTHESIZED REGULAR EXPRESSIONS • INCREMENTAL/GLOBAL SEARCH AND REPLACE ACROSS ALL BUFFERS • **BLOCK COMMANDS** • COLUMN/LINE/STREAM BLOCKS • SAVE • CUT • DELETE • BLANK • SEARCH-FOR • FORMAT • JUSTIFY LEFT/RIGHT/CENTER • UPPER/LOWER CASE • REMOVE • OVERLAY • REACTIVATE • TAB • DRAG • SORTS • SPREADSHEET STYLE MATH - ADD/SUB/MULT/DIVD/AVG • **MACROS** • MENU DRIVEN • VIEW • AUTOEXECUTE • TIME-DELAY • NESTED • RECURSIVE • INTERACTIVE• 1 OR 2 KEYSTROKE • SCREEN DISPLAY CONTROL • SIZE LIMITED BY AVAILABLE MEMORY • AUTOMATIC MACRO MENU CREATION • **MORE FUNCTIONS** • PULLDOWN USER MENU SYSTEM • RECONFIGURE COLOR AND WINDOW STYLE • VIEW/MANIPULATE BUFFERS THROUGH MENU OR KEYSTROKE • INTELLIGENT DIAGRAM/BOX DRAW • ADJUSTABLE SCROLL VALUE • TRANSPOSE LINES/CHARACTERS/WORDS • MOVE-TO-POSITION STACK • VISUAL TAB CREATION • POPUP RULER • COLUMNAR/ENTIRELINE TABS • TIME AND DATE STAMP • RESTORE TYPED-OVER CHARACTERS • CHANGE DIRECTORY • BRACKET/BRACE/PARENTHESIS MATCHING • ENTER GRAPHICS CHARACTERS • EXCUTE DOS COMMANDS • POPUP ASCII TABLE • USER DEFINED POPUP FILES • KEYBOARD CONFIGURATION • PRINT BUFFER WITH PAGE NUMBERS, ADJUSTABLE SPACING AND MARGINS • INTERACTIVE TUTORIAL • MANUAL AND TUTORIAL ON-LINE • WRITTEN IN C AND ASSEMBLER • **EXTERNAL UTILITIES** • INTELLIGENT CHANGE DIRECTORY COMMAND • POPUP DIRECTORY FACILITY • STRING TRANSLATOR • **HARDWARE REQUIREMENTS** • DOS • PC/XT/AT, PS/2, 386 • CGA, MDA, EGA, HERCULES, WYSE 700/AMDEK 1280 • RUNS IN ALL VIDEO MODES, NO FLAGS, NO DRIVERS • 256K • **FOR LITERATURE AND DEMO:**

CALL 201-450-4545

VERSION 1.0

PRICE: $265.00

MASTERCARD, VISA, C.O.D, AND P.O.'S

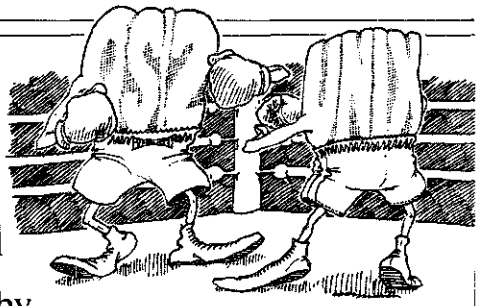
THE AMERICAN COSMOTRON 80 HOLMES ST • PO BOX 128 • BELLEVILLE, NJ 07109

Number **215** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

OS/2 vs. Unix

The Highly Touted and the Much-Maligned



by
Frank D. Greco

Our Unix correspondent has steeped himself in OS/2 these many months, and in this first of a series makes an admirably even-handed comparison of the ways in which OS/2 and Unix handle the issues of multitasking, multiusers, filesystems management, and interprocess communication. In subsequent articles Frank will look at OS/2 and Unix in light of the other issues on his list—shell interfaces, graphics, networking, windowing, portability, and "future direction."

"We're Not in Kansas Anymore"

If you need to take advantage of the new muscle processors—the 80386/486, 68030/040, and SPARC—a multitasking operating system is the obvious choice. If you're a professional programmer, you're probably weary of the limits of single-tasking MS-DOS and long for the direct benefits of concurrent operating systems—asynchronous I/O, simultaneous numerical computations, computer simulation, window systems, and probably most important, network applications. Early or late, you've probably realized that if these activities are to go on cleanly and elegantly, you have to move away from the simple-minded DOS world. True, most of you have been doing MS-DOS programming for quite a while, simulating multitasking with TSRs, but that's serial multitasking; the interrupted task is suspended! Let's face it, true multitasking operating systems *are* useful (I can't believe I still have to say this in 1988!), and if you are not prepared for it, one of these days a whirlwind of events is going to pick you up and plop you right down in unknown tasking territory.

Two of the most important current multitasking operating systems are Unix and OS/2. Unix is about twenty years old and is enjoying a resurgent popularity with the help of Sun Microsystems and the mysterious motives of the nonprofit (ha!) Open Systems Foundation (OSF). OS/2 is the new kid on the block, borrowing concepts popularized in Unix. As yet unproved in the end-user community, OS/2 is touted as the next Mike Tyson of the computer industry. These two operating systems are fighting for development dollars. Let's investigate these two operating systems in detail to determine their differences and similarities.

I've separated the operating system issues into the following categories:

- Multitasking
- Multiusers
- Filesystems
- Interprocess Communication
- Shell Interfaces
- Graphics
- Networking
- Windowing
- Software Portability
- Compatibility with MS-DOS
- Future Direction

Multitasking

Multitasking is of course the ability to simultaneously execute more than one task, and that breaks down into three basic types: serial multitasking, multiprocessor multitasking, and true multitasking. In serial multitasking you can interrupt a currently executing task (for example, a text editor), suspend it, execute another task (SideKick, for example), then come back to the original task and pick up where you left off. Terminate and Stay Resident programs under MS-DOS are examples of this one-at-a-time serial multitasking. In multiprocessor multitasking, separate processors are assigned to execute a task or a portion of a task. Neither OS/2 nor Unix is adequately designed to handle fine-grained or coarse-grained multiprocessing.

True multitasking (a misnomer for sure) is what we are concerned about. In true multitasking a single processor is time-multiplexed to give the appearance of parallelism. Both OS/2 and Unix offer this type of multitasking. A true multitasking operating system demands more from the hardware underneath than a simple, single-tasking OS does. When several concurrent processes are running under Unix or OS/2, each process steals cycles from the other using the timer interrupt. The result is that while the overall performance of the computer system is enhanced, the response time for a particular application may appear to be degraded. It's surprising how many end-users don't understand that you don't get something for nothing.

Upper Management

Multitasking under OS/2 can be broken down into three sections: Session (or Presentation) Manager, processes, and threads. At the highest level is the Session Manager or Presentation Manager. When one or more OS/2 processes share the same virtual console (screen, keyboard, and mouse), this group of processes is called a *screen group*. In the absence of the Presentation Manager, the Session Manager is the software that manages the switching of these screen groups. The Presentation Manager is a graphical replacement for the session manager and performs window management. You can think of the Session (or Presentation) Manager as the user level control of processes and threads.

A *process* under OS/2 has the same definition as its Unix counterpart. It's an executable image of a program; an instance of a running program. In hierarchical fashion, an OS/2 process inherits much of its environment from its parent process. Resources such as open files, process priority, devices, screen group, memory, etc., all form the parent process's legacy to the child process. See Listing 1 for a simple example of spawning a child process using the OS/2 **DosExecPgm()** system call.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Threads of Evidence

The scheduler in OS/2, similar to Unix's, is a priority-based, preemptive scheduler. In OS/2, however, the "dispatchable entity" is called a *thread* as opposed to a *process*. This implies that each schedulable thread has a corresponding priority. There are four basic priority categories for threads (time-critical, foreground, regular, and low), all with varying numeric levels of priority. You can instruct OS/2 at boot time (via an entry in config.sys) to dynamically adjust the priority of regular class threads, or alternatively, to keep a process at a certain priority level with the regular class. And unlike Unix's scheduler, the OS/2 scheduler itself is not a schedulable entity. Checkpoints in the kernel of OS/2 allow a determination whether another, higher, priority, thread should run. Apparently, the feeling at Microsoft is that a time slot for the OS/2 scheduler impacts the overall performance of the system by stealing cycles for operating system administrative overhead.

Can Threads Be Used for Spooling?

Basically, there are two major reasons for using threads instead of processes: for fast, cheap (minimal resource spending) multitasking within a process and for the ability to easily share data among cooperating concurrent entities. There can be several threads, each with its own thread identification number (thread ID), running through a given process. A process has at least one thread, but can have many more. By comparison, most normal MS-DOS programs have only one thread of execution. Threads are yet another resource owned by a process. The default number of threads (alterable by another entry in config.sys) is 64, which is allocated by giving 24 to the OS/2 kernel and 40 to application programs.

When a user runs an application program, that is, when the application has been placed in the currently active screen group visible to the user, the priority of the process interacting with the virtual keyboard is raised to foreground. This insures fast response to the end user. Under the Unix operating system, however, the priorities of processes are not changed because a process is put into the foreground or background (some shell programs may change the priority of a process when starting a background process, though). Priorities can always be adjusted lower by the user using the "nice" or "renice" utilities (see Listing 2). Only the so-called Unix superuser can adjust priorities higher. However, because it is designed for "fair scheduling," Unix always dynamically alters the priorities of processes so that no one process hogs the CPU. The priorities of CPU-intensive Unix processes are dynamically lowered over time so that I/O-based processes get a chance at the CPU. The OS/2 programmer, on the other hand, can control even timeslicing by setting a YACV (Yet Another Config.sys Variable) called TIMESLICE; TIMESLICE can be set at from between 32 to 248 milliseconds. All in all, the OS/2 programmer has a finer degree of control over process/thread management than the Unix programmer. In practice, however, it has yet to be determined whether the different thread priority categories are more complex than they need to be.

Unix: Don't Thread on Me

Threads, or lightweight processes (LWPs), is an old concept that harks back to the simple days of a single address space. Languages such as Modula-2, Mesa, and Simula have all incorporated lightweight processes. There are several interesting Unix implementations such as the one documented in the Summer 1985, Usenix paper, "Lightweight Processes for Unix—Implementation and Applications" by Jonathan Kapecs of Sun Microsystems. Doug Comer describes a simple and cheap context switcher for the LSI 11/02 in his Xinu book. Another implementation was described in the Usenix Association newsletter Vol.12, No.4, although not so clearly, by Don Libes. In addition, dozens of lesser implementations (an apology to my undergraduate students is in order here) have been performed using BSD Unix's **setitimer()** and **setjmp/longjmp**.

Sun's implementation of LWPs (first documented in Jonathan Kapecs' paper) is formally available in the most recent release of the Sun operating system, SunOS 4.0. As AT&T has publicly announced, most of the features of SunOS 4.0 (and SunOS 4.1) will be incorporated into AT&T System V.4. This version of Unix will be available in the last quarter of 1989. I should point out that the OS/2 kernel (and I take some liberties in calling it a kernel . . .) offers support for threads while the Unix, that is, SunOS kernel (ditto), currently does not. This implies that concurrent system calls are not allowed under Unix (they must be serialized), and a LWP that uses blocking I/O might monopolize the CPU for the entire time slice of the process. Sun has said that kernel support for LWPs will be added in an upcoming release. A work-around involves linking a non-blocking I/O library provided by Sun. On the other hand, threads are deeply rooted in the OS/2 kernel. As a matter of fact, the OS/2 scheduler does not even know about processes, just about threads.

Multiusers

A multiuser operating system allows two or more concurrent users to execute processes simultaneously.

The current trend in the industry, in case you haven't noticed, is to connect single-user multitasking workstations together to form a distributed networked environment. Just as it can easily support a single user, Unix can also be used as a multiuser operating system. To Unix, multiuser access is just about the same as single user access. The only real difference, besides AT&T license fees, is some entries in a couple files that determine whether or not to create a login (for example, **userid**) and turn on terminal access (spawn a **getty**). In fact, multi-user-ness in Unix is almost the same as multitasking-ness. The only difference is a bunch of ASCII files consulted at certain times while a Unix system is alive.

On the other hand, Bill Gates has made it known that Microsoft wants to push single-user, OS/2-powered, networked workstations as their vision of the future. Multiuser-ness is anathema to the Redmond Rogues. But at a recent seminar on OS/2 sponsored by Microsoft, someone in the audience asked if OS/2 was capable of supporting several concurrent users. The answer from a Microsoft representative was "there is

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

nothing in OS/2 that prevents multiuser access." I don't think Microsoft is that interested in multiuser OS/2, but I have heard strong rumors that IBM is planning to take OS/2 multiuser (a not too atypical IBM move, I might add).

Filesystems

In this context, filesystems are basically split into two parts: the files themselves on the secondary storage devices and how the operating system manages them.

Duplicate File (Concept) Found

OS/2 is designed so that every object is defined in terms of the filesystem name space. In other words, everything is considered a logical file. Files, shared memory, semaphores, serial ports, named pipes, etc.—sound familiar? They've been prime features of the Unix operating system since its inception. The only difference is that the current release of OS/2 does not have real filesystem entries for these objects; Unix actually does. In Unix parlance, there is an *inode* for devices, but there are no data block indices in the inode (that is, there are no data). Instead, the major and minor numbers representing the device are stored in the data blocks index array.

Much research has gone into the Unix filesystem since the early days, and with its enhancements, it's very mature, secure, robust, and fast. Filesystem performance has been increased externally by smarter shell interfaces (caching, hashed access, etc.). With the advent of LWPs, we are going to see even faster filesystems under Unix, especially daemon-based filesystems like NFS.

Filenames themselves can be up to 256 characters in Berkeley-based Unixes and do not have required filename extensions such as OS/2's .EXE, .BAT, and .CMD. (The three-character extension is a vestigial naming convention bequeathed by an old DEC operating system.) Unix has only conventional filename extensions, such as .c, .out, and .sh.

Install Tactic

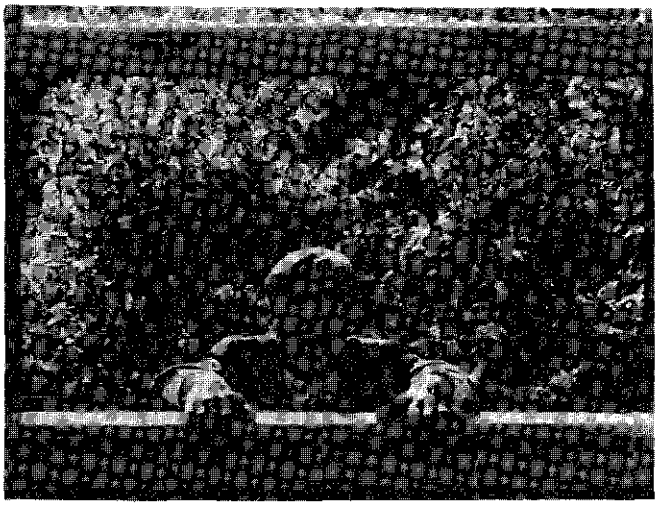
With a lot of foresight, the OS/2 designers didn't build the filesystem into OS/2. In the future there may be storage devices that the current OS/2 can't handle, so the filesystem has been designed to be installable. When a foreign filesystem for an unusual storage device is built, it would just be installed and integrated into OS/2 and accessed in the same manner as other devices. The current implementation of Unix does not have "installable" filesystems, per se. The acyclic directed graph (commonly, but erroneously, called a tree) filesystem is deeply ingrained in the core of Unix. (This is one reason Unix is not a true kernel; a true kernel dispatches filesystem requests to an external entity.)

A Unix equivalent to Microsoft's installable filesystems (IFS) is the Virtual File System (VFS) and, to an extent, the Network File System (NFS) concepts created by Sun Microsystems. The abstract VFS interface, while not installable, lives in the Unix kernel and allows foreign filesystems (DOS filesystems, VMS filesystems, optical disk filesystems, NFS filesystems, etc., for example) to coexist with normal Unix filesystems.

File Name Space Transparency Is Hard to See

I'm sure that the success and elegance of Sun's NFS product has influenced the design of the OS/2 filesystem. Microsoft has said that accessing a remote logical file over the network through Microsoft's OS/2 LAN Manager will be

Does your windowing package leave you hanging?



Frustrated with the limitations of your windowing package?

ÆWINDOS is a high-powered set of tools designed to interactively create windows at light speed with both DOS and UNIX systems.

Written entirely in portable "C", it includes the complete source code and even make files, so changes are quick and easy.

Don't wait until you're caught in a bind, call us today. Modular, portable, and efficient, ÆWINDOS won't leave you hanging.

ÆSOFT

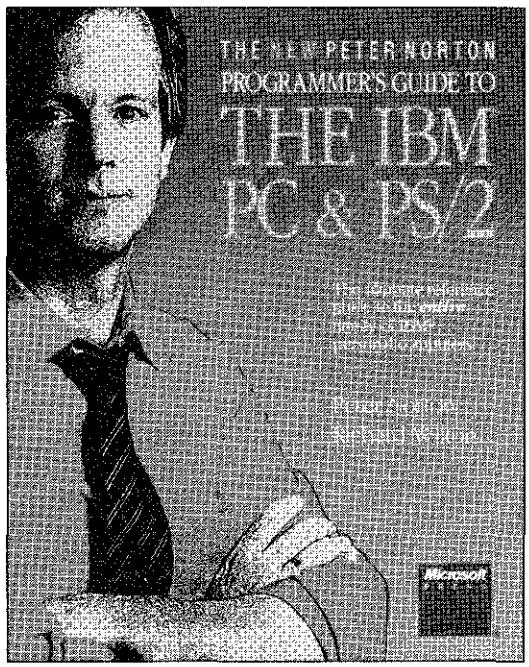
1-800-634-5494 BBS (303) 449-5714

Number 220 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Peter Norton's Best

"... both a text to be read from front to back and a reference book to sit on the shelf waiting with immediate answers to questions ... a must for every serious PC user's technical library." ***PC Magazine***



$22.95

THE PETER NORTON PROGRAMMER'S GUIDE TO THE IBM PC has long been a must-have reference for programmers, students, industry experts, even nonprogrammers who want to master the inner workings of IBM microcomputers. Now thoroughly rewritten, THE NEW PETER NORTON PROGRAMMER'S GUIDE TO THE IBM PC AND PS/2 continues to be *the* reference of choice to the new generation of IBM microcomputers—including the PS/2. Accept no substitutes.

Timely New Information

A wealth of updated and new information covers 80286 and 80386 microprocessors; the new PS/2 ROM BIOS; the enhanced keyboard; the new VGA and MCGA; DOS through version 4.0; interrupts, device drivers, and video programming; and programming in C, Microsoft QuickBASIC, and Turbo Pascal.

Unmatched Programming Advice

Norton and Wilton's characteristically astute advice, programming methods, and techniques will help you create simple, clean programs that are portable among all current—even future—IBM machines. You won't find better authorities—their down-to-earth style makes even the most complex concepts readily understandable.

Solid Technical Data

Included is up-to-date information on ROM software; video, disk, and keyboard basics; clocks, timers, and sound generation; ROM BIOS basics and the ROM BIOS services; DOS basics, interrupts, and functions; and much more. Of special interest are the details of the architectural similarities and differences among IBM machines.

Microsoft® Press

Hardcore Computer Books

Available wherever books and software are sold.
Or call in your credit card order ***1-800-638-3030*** *(8AM to 4PM EST).*

Number 234 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

It's Our Objective To Keep You Oriented

NEW

JOURNAL OF OBJECT-ORIENTED Programming

Editor: **Dr. Richard Wiener** *Univ. of Colorado*
Assoc. Editor: **Prof. David Thomas,** *Carleton Univ.*

An innovative bimonthly journal offering original and stimulating research in all areas of object-oriented programming.

The Journal of Object-Oriented Programming (JOOP) is designed to track and evaluate the advances in the object-oriented methodology.

Well-researched articles on topics pertinent to the field and your daily work will appear, such as:

- **problem-solving techniques**
- **reuseable OOP components**
- **latest language developments**
- **applications in artificial intelligence**
- **software maintenance**
- **unbiased product reviews**
- **up-to-the minute product news**

JOOP maintains a prestigious international editorial board from industry and academia, which include:

Adele Goldberg
ParcPlace Systems
Brad Cox
StepStone Corp.
Bjarne Stroustrup
AT & T Bell Labs.
Kurt Schmucker
Apple Computer

The journal's incisive articles are written by well-known academic researchers and premier software developers from around the world. Important upcoming articles include:

"Active Objects, An Access Oriented Framework for Object-Oriented Languages" Dr. Kasper Osterhye

"The Model View Controller Cookbook" Dr. Stephen Pope

"Using Objects to Package User Interface Functionality" Kurt Schmucker

"Designing Reusable Classes" Dr. Ralph Johnson
"Smalltalk-80 Based Computer Music Tools" Dr. Stephen Pope

"Communication, Scheduling and Resource Management in SINA—A New Object-Oriented Language" Dr. Armand Tripathi

JOOP is written for software developers, engineers, researchers and computer scientists. Get original research that's definitive, authoritative and applicable. Get objective coverage that's enlightening, reliable and functional.

Subscribe Today

Order toll free **1-800-345-8112** (7 days a week)
or mail in this coupon

Yes, send me JOOP at the following term

1 year (6 issues)
☐ individual - $49 ☐ institutional - $79
2 years (12 issues)
☐ individual - $89 ☐ institutional - $146

Order outside the U.S., add $24 a year for Air Service.

Check enclosed _____ Bill me _____
Charge my: Visa _____ Mastercard _____
Card # _____ Exp. Date _____
Signature _____

Your Name _____
Company _____
Address _____
City _____ State _____ Zip _____

Individual subscriptions must be paid with a personal check.
Make checks payable to **SIGS Publications**
Mail to: JOOP, 310 Madison Avenue, Suite 503, New York, New York 10017

Code 88

Number 205 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

as transparent as using a local file. Aside note: Microsoft has said that support for shared memory across a network will not be supported; it is too slow. Obviously, with the current 1, 2, and even 10 megabits per second, it *is* too slow. Given a faster network, I wonder if Microsoft would think of the possibilities of distributed debugging using shared memory. Sun Microsystems' Bill Joy has said that networkable shared memory, though costly over standard ethernet, will be available from Sun once Fiber Distributed Data Interface (FDDI) networks are installed en masse. The Joy of Sun has projected the end of 1989 as the start date for 100+ megabit per second networks.

Active Files

Another thought—I wonder how easy it will be to implement Tom Killian's "Processes as Files" concept under OS/2. Tom is from AT&T Bell Labs, and his elegant idea is over four years old. Most certainly, OS/2 programmers will find that debugging unrelated processes in an intrinsically hierarchical process spawning model such as can be found in both OS/2 and Unix can be frustrating. The **ptrace()** mechanism available in both operating systems (in typical fashion, OS/2 calls it **DosPtrace**) can trace a child process only. Actually, in OS/2, the process being debugged does not have to be a direct child of the process calling **DosPtrace**; the last process execed using the trace flag of the **DosExecPgm** call is the one that will be debugged. Many times it is desirable to debug a totally unrelated active process. Killian's idea is to make active Unix processes part of the filesystem (that is, "/proc/pidno"). The concept of having active processes as file entities allows all sorts of innovative debugging methodologies. Using this concept and the transparency of NFS, a debugger can inspect and change (with appropriate file permissions, of course) active processes all over the network. Cute idea and a damn powerful one at that.

Will You Still Love Me, Will You Still Need Me, When I'm 64?

One unfortunate aspect of the current OS/2 filesystem is that it still uses the old-fashioned MS-DOS FAT-oriented filesystem. (And you thought that Time Marches On . . .) Cluster chains (linked lists) are still used, and users still need to name the disk drive by letter. In Microsoft's defense, I should add that enhanced filesystems will be available in a future release of OS/2.

Interprocess Communication

Interprocess Communication (IPC) is a way for multiple processes to communicate information. Despite the fact that OS/2 threads can communicate with other threads using the OS/2 IPC mechanisms, it is still called Inter*process* communication.

OS/2 has several IPC mechanisms:

- Signals
- Named Pipes
- Anonymous Pipes
- Shared Memory
- Semaphores
- Messages and Message Queues
- Dynamic Data Exchange

Unix has several IPC mechanisms:

- Signals
- Named Pipes
- Un-named Pipes
- Shared Memory
- Semaphores
- Messages and Message Queues
- Streams
- Sockets

To make the lists really complete, we should add files and command line arguments as coarse ways of communicating between processes.

Signals, both types of pipes, shared memory, semaphores, and messages are, more or less, the same in both operating systems. Unix, however, has an advantage in that it offers both the sophisticated Streams and the Sockets mechanisms. OS/2 needs to come up with a method at least equivalent to the Unix socket to perform local and remote interprocess communication.

I'll leave a detailed explanation of these different types of interprocess communications for a future article. But I just can't stop myself from explaining what IBM and Microsoft tout as their greatest communications paradigm, the named pipe. Regular (or anonymous) pipes are a communications medium between related processes. Named pipes (or FIFOs) are used to send data between unrelated processes. FIFOs were implemented in Unix System III a long time

FULL AT&T C++: ANNOUNCING VERSION 1.2

Guidelines announces its port of **version 1.2** of AT&T's C++ translator. As an object-oriented language C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. "Object-oriented" means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2.** C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: $295

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.
Note: C++ is a *translator,* and requires the use of Microsoft C 3.0 or later.

Here is what you get for $295:
- The full AT&T v1.2 C++ translator.
- Libraries for stream I/O and complex math.
- **The C++ Programming Language**, the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Improved installation guide and documentation.
- 30-day money-back guarantee.

To Order:
send check or money order to:

GUIDELINES SOFTWARE, INC.
P. O. Box 749, #PJ
Orinda, CA 94563

To order with VISA or MC, phone (415) 254-9183.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

Circle number **204** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ago. Believe it or not, messages, shared memory, and semaphores actually were in an internal version of Unix long before FIFOs were implemented on Unix (we're talking circa 1978). Just to give you a taste of named pipes, I've written some real quick and dirty C code (Listings 3, 4, 5 and 6) to demonstrate the ease of using named pipes under Unix.

Window to Window

As far as interprocess communication protocols between windows are concerned, the OS/2 Presentation Manager is one step ahead of its Open Look opponent. The Dynamic Data Exchange (DDE) is a protocol specification that allows data to be exchanged between two processes in windows. DDE originally came to life in Microsoft's Windows product a few years back. Although Microsoft's DDE specification detailed only the shuttling of data between windows on a single machine, a third party vendor has extended DDE to work over a network (that's what it should have done in the first place!). The Open Look specification, as of July 1988, does not discuss any interwindow communications protocol at all, a deficiency that will have to be corrected if we are to obtain the so-called "hot link." This is not a breakfast food; it is the ability of a change in one program to trigger instant changes in other programs. For example, a change in a spreadsheet may trigger an email message to one of your clients in New York, an update to a database, and a different bitmap to be included in a publishing package. Light Communication

Unix's lightweight processes, or OS/2's threads, also have another way of talking to other processes/threads within a given process. By definition, all the threads (LWPs) that belong to a (heavyweight) process share the same address space. That is one of the biggest advantages of LWPs. Sometimes the memory protection mechanisms between processes get in the way and we have to use some sort of communication mechanism to share variables. In many cases this is an unnecessary added complexity.

Start From Scratch

The disadvantage of the Unix System V IPC mechanisms is that they are not that well integrated into the Unix kernel. They were "hacked on" post priori to suit the customers of AT&T Unix. Additionally, support for multiprocessor IPC is almost non-existent in either Unix or OS/2. An operating system should be built with IPC primitives to start off from; any future operating system must include networking and multiprocessor IPC mechanisms. Rewriting the Unix kernel from scratch and maintaining the Application Binary Interface (ABI) and Application Programming Interface (API) is certainly a step in the right direction. ♦

To be continued

Frank D. Greco is an independent consultant who does industry analysis, systems consulting, and technological forecasting for Fortune 500 companies. Readers can contact Frank c/o Crossroads Technologies, Inc., P.O. Box 530, Fanwood, NJ 07023, (201) 754-7820.

Code follows.

DEVELOPMENT TOOLS FOR C

- A COLLECTION OF "TOOL BOXES" FOR THE DEVELOPMENT OF POWERFUL C PROGRAMS
- DESIGNED TO BE USED BY BOTH NEW AND EXPERIENCED C PROGRAMMERS
- EACH TOOL BOX IS FULLY DOCUMENTED -- WITH EXAMPLES

SYSTEM TOOL BOX
- ACCESS THE POWER OF YOUR PC
- KEYBOARD & SCREEN MANAGEMENT
- GRAPHICS & MUCH MORE
- 184 PAGE USER'S GUIDE

LIB + SRC $35

UTILITY TOOL BOX
- GENERAL PURPOSE FILTERS
- UTILITY PROGRAMS
- GENERAL PURPOSE FUNCTIONS
- 86 PAGE USER'S GUIDE

LIB + SRC $25

WINDOW TOOL BOX
- COMPLETE WINDOW CONTROL
- BUILD WINDOW DATABASES
- FULL ERROR REPORTING
- 84 PAGE USER'S GUIDE

LIB $45 -- SRC $125

MENU TOOL BOX
- VERTICAL/HORIZONTAL MENUS
- CREATE MENU DATABASES
- FULL ERROR REPORTING
- 78 PAGE USER'S GUIDE

LIB $35 -- SRC $75

- HARDWARE -- IBM PC, TI PRO OR TANDY 1000
- COMPILER -- MICROSOFT™ C (3.00 & LATER)

- CALL OR WRITE FOR FREE INFO & DEMO PROGRAM
- TECHNICAL SUPPORT IS UNLIMITED & FREE
- NO ROYALTIES

FLEMING SOFTWARE
5741 Wagonwheel Trail
Castle Rock, CO 80104
(303) 688 - 3584

Circle number 118 on the Reader Service Card

PCYACC™
Version 2.0

PCYACC Version 2.0 is a program generator capable of automatically generating ANSI C source code for building Assemblers, Compilers, Browsers, Page Description Languages, Language Translators, and Query languages.

Complete grammars and lexical analyzers for ANSI C, C++, ISO Pascal, DBASE III/Plus, SQL, Smalltalk, and YACC are included.

Example application sources are provided to be used as skeletons for new programs. Examples include a desktop calculator, an Infix to Postfix Translator, a DBASE III/Plus Syntax analyzer, an implementation of the PIC[ture] language, and a C++ to C translator.

PCYACC is input compatible with UNIX YACC when used with C grammar descriptions.

- Quick Syntax analysis option
- Optional Abstract Syntax Tree
- Advanced Error Recovery Support Provided
- All examples include FULL SOURCE listings
- Manual "Compiler Construction with PC'S" included
- 30 day money back guarantee

Professional version $395.00

ABRAXAS SOFTWARE, INC.
7033 SW Macadam Ave.
Portland, OR 97219
(503) 244-5253
FAX (503) 245-1110

Circle number 227 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk.
ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386.
Mail order information on pages 8 and 88.

Code from "OS/2 vs. UNIX" by Frank D. Greco

1

Listing 1

```
/*
 * dosexec.c     -----    Spawns a synchronous OS/2 process
 */

#include <doscall.h>
#include <stdio.h>
#include <string.h>

#define WAIT                (unsigned) 0
#define NO_WAIT_DONT_CARE   (unsigned) 1
#define NO_WAIT_CARE        (unsigned) 2
#define PTRACE              (unsigned) 3
#define DAEMON              (unsigned) 4

char FailString[64];                /* Place for name of load object on error */
char far *FSptr = FailString;       /* Ptr to Fail String buffer */
unsigned FSLength = sizeof FailString;  /* Length of failure string */
unsigned ExecType;                  /*  Wait,
                                        NoWaitDontCare, NoWaitCare,
                                        Ptrace,
                                        Daemon */
char ArgString[64];                 /* Argument string */
char far *ASptr = ArgString;        /* Far ptr needed for DosExecPgm() */
char EnvString[64];                 /* Environment Space string */
char far *ESptr = EnvString;        /* Far ptr needed for DosExecPgm() */
char Pgm[64];                       /* Program to be exec'ed */
char far *Pptr = Pgm;               /* Far ptr needed for DosExecPgm() */

struct ResultCodes RetCode;         /* For return code from DosExecPgm() */
struct ResultCodes far *RCptr = &RetCode;

main()
{
    char *offset;

    strcpy(Pgm, "frank.exe");       /* Want to  run 'frank.exe' */
    ESptr = (char far *) NULL;      /* Inherit Parent's environment */

    strcpy(ArgString, "frank.exe"); /* First string is pgm name */

    offset = ArgString + strlen("frank.exe");
    *offset = NULL;                 /* Separate the two strings */
    strcpy(offset+1, "-d file");    /* Second string = the cmd line arguments */

/* Now that everything is setup...go and do it. */

    DosExecPgm(FSptr, FSLength, WAIT, ASptr, ESptr, RCptr, Pptr);
}
```

Listing 2

```
/*
 *  n.c     -----   Runs a Unix process at a desired "nice" value.
 */

#include <stdio.h>

main(argc,argv)
char **argv;
{
    int nicevalue;      /* Desired nice value */
/*
 * A real program should perform decent error handling.  This one
 * does not.
 */
    nicevalue = atoi(argv[1]);  /* Get nice value from cmd line */

    nice(nicevalue);        /* Set nice value */
    execvp(argv[2], &argv[2]);  /* Run desired pgm using nice value*/
}
```

2

Listing 3

```
/*
 * daemon.c     -----   The background process that waits for events to happen
 *                      and performs some task based on that event.  Most
 *                      of the time, a daemon's action goes usually unnoticed
 *                      by users.
 */

#include <sys/signal.h>
#include <fcntl.h>
#include <stdio.h>
#include "local.h"

/*
 *  This is not the most complete method of creating a Unix daemon.
 *  Please see the paper by Dave Lennert in the Usenix ;login: newsletter
 *  dated July/August 1987 called "How To Write a Unix Daemon".
 */

main(argc,argv)
char **argv;
{
    int fd;
    PACKET test;

    signal(SIGINT, SIG_IGN);    /* Only kill self when client says so */


/*  Now open the FIFO for reading */

    if ( (fd = open(PIPE, O_RDONLY))  ==  -1 )
        fprintf(stderr, "Cannot open PIPE\n"), exit(1);

    while(1)
    {
        read(fd, &test, sizeof test);           /* Read client via FIFO*/

        if ( !strcmp(test.string, "Done") )
            printf("Byebye from Daemon\n"), exit(1);

        printf("PID: %d - string: [%s]\n", test.pid, test.string);
    }
}
```

Listing 4

```
/*
 * client.c         -----   A potential customer for the background process
 *                          (i.e., "daemon").  All it does is send a string
 *                          and its process id to the daemon.
 *                          Note that read() and write() are atomic,
 *                          uninterruptable system calls.  This guarantees
 *                          that our data arrive in one piece and not
 *                          interleaved with some other client's data.
 */

#include <stdio.h>
#include <fcntl.h>
#include <sys/signal.h>
#include "local.h"

int fd;     /* File descriptor for the Named Pipe (FIFO) */
PACKET msg; /* Information to be sent over a FIFO to a server process */

main(argc,argv)
char **argv;
{
    int quit();                         /* What to do when killed */

    srand(getpid());

    msg.pid = getpid();                 /* Setup the data to be sent */
    strcpy(msg.string, "Message from a Client");


/* Open the FIFO (Named Pipe) for writing */

    if ( (fd = open(PIPE, O_WRONLY))  ==  -1 )
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Greco Code, Continued

```
    fprintf(stderr, "Cannot open PIPE in client\n"), exit(1);
signal(SIGINT, quit);                   /* Prepare for a Break */

while(1)
{
    write(fd, &msg, sizeof msg);        /* Write to server */
    sleep(rand() % 10);                 /* Nice and slow, so we can see msgs */
}
}
/*************************************************************/
quit()
{
    strcpy(msg.string, "Done");         /* Tell server to kill itself */
    write(fd, &msg, sizeof msg);
    exit(0);                            /* Kill this particular client */
}
```

Listing 5

```
#define STRMAX      32
#define PIPE        "./FrankPIPE"

typedef struct _PACKET
{
    int pid;
    char string[STRMAX];
} PACKET;
```

Listing 6

```
---- Running the named pipe program in Unix -----

% mknod FrankPIPE p              # This makes the named pipe in Unix
                                 # I'm just using the arbitrary filename 'FrankPIPE'

% ls -l                         # Note the 'p' for (named) pipe for FrankPIPE
total 92
prw-rw-rw-   1 frank    1000        0 Aug 27 14:39 FrankPIPE
-rwxrwxrwx   1 frank    1000    16836 Aug 27 14:34 client*
-rw-rw-rw-   1 frank    1000      912 Aug 27 14:34 client.c
-rwxrwxrwx   1 frank    1000    15828 Aug 27 14:36 daemon*
-rw-rw-rw-   1 frank    1000      549 Aug 27 14:36 daemon.c
-rw-rw-rw-   1 frank    1000      110 Aug 27 14:24 local.h

% daemon > logfile &
284                              <-------------------- The shell prints the pid of the daemon process

% client & client & client &
285 <-------------------- The shell prints pids of the 3 client processes
286
287

% sleep 10; cat logfile
PID: 286 - string: [Message from a Client]
PID: 285 - string: [Message from a Client]
PID: 287 - string: [Message from a Client]
PID: 286 - string: [Message from a Client]
PID: 287 - string: [Message from a Client]
PID: 286 - string: [Message from a Client]
PID: 286 - string: [Message from a Client]
PID: 287 - string: [Message from a Client]
PID: 285 - string: [Message from a Client]
PID: 286 - string: [Message from a Client]
PID: 285 - string: [Message from a Client]
PID: 286 - string: [Message from a Client]
PID: 285 - string: [Message from a Client]
PID: 287 - string: [Message from a Client]
PID: 287 - string: [Message from a Client]
PID: 286 - string: [Message from a Client]
                                 <-------- I sent a SIGINT signal to a client ◆
Byebye from Daemon
```


"If you need source code, make sure your wallet is wide open or get VITAMIN C."
- Computer Language, June, 1987

Vitamin C

It's good for your system!

- Professional C function library
- 30 day money back guarantee
- Multiple bullet proof windows
- Easy full screen data entry
- Unlimited data validation
- Context sensitive help manager
- Menus like Lotus and Mac
- Programmable keyboard handler
- Text editor routines
- No royalties or runtime fees
- Library source included FREE
- Free technical support
- Free BBS at (214)418-0059
- Supports all major compilers including Microsoft 5.0
- VCScreen code generator too!
- UNIX version avaialable

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination that software professionals demand. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

Vitamin C $225
Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.

VCScreen $99.95
For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.

ORDER NOW!

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

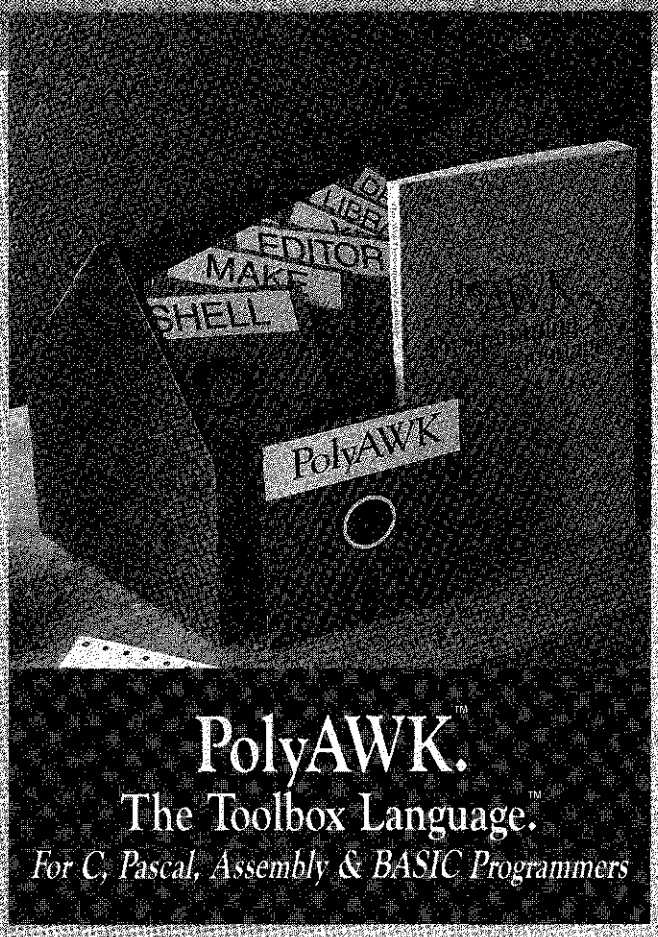
(214)416-6447

creative PROGRAMMING
Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



PolyAWK.™

The Toolbox Language.™

For C, Pascal, Assembly & BASIC Programmers

MS-DOS $99

We call PolyAWK our "toolbox language" because it is a general-purpose language that can replace a host of specialized tools or programs. You will still use your standard language (C, Pascal, Assembler or other modular language) to develop applications, but you will write your own specialized development tools and programs with this versatile, simple and powerful language. Like thousands of others, you will soon find PolyAWK to be an indispensable part of your MS-DOS toolbox. If you don't have enough OS/2 development tools, now it's easy to write your own.

A True Implementation Under MS-DOS & OS/2

Bell Labs brought the world UNIX and C, and now professional programmers are discovering AWK. AWK was originally developed for UNIX by Alfred Aho, Richard Weinberger & Brian Kernighan of Bell Labs. Now PolyAWK gives MS-DOS & OS/2 programmers a true implementation of this valuable "new" programming tool. PolyAWK fully conforms to the AWK standard as defined by the original authors in their book, *The AWK Programming Language*.

A Pattern Matching Language

PolyAWK is a powerful pattern matching language for writing short programs to handle common text manipulation and data conversion tasks. A PolyAWK program consists of a sequence of patterns and actions that tell what to look for in the input data and what to do when it's found. PolyAWK searches a set of files for lines matched by any of the patterns. When a matching line is found, the corresponding action is performed. A pattern can select lines by boolean combinations of regular expressions and comparison operations on strings, numbers, fields, variables, and array elements; patterns may reference properties of the current input line or any other program variables. Actions may perform arbitrary processing on selected lines. The action language looks like C, but there are no declarations, and strings and numbers are built-in data types. You can have multiple input files and output files, regular expressions, user-defined functions, and run other programs.

Saves You Time & Effort

The most compelling reason to use PolyAWK is that you can literally accomplish in a few lines of code what may take pages in C, Pascal or Assembler. Programmers spend a lot of time writing code to perform simple, mechanical data manipulation — changing the format of data, checking its validity, finding items with some property, adding up numbers and printing reports. It is time consuming to have to write a special-purpose program in a standard language like C or Pascal each time such a task comes up. With PolyAWK, you can handle such tasks with very short programs, often only one or two lines long. The brevity of expression and convenience of operations make PolyAWK valuable for prototyping even large-sized programs.

Very Concise Code

Where program development time is more important than run time, AWK is hard to beat. These AWK characteristics let you write short and concise programs:

- The implicit input loop and the pattern-action paradigm simplify and often entirely eliminate control flow.
- Field splitting parses the most common forms of input, while numbers and strings and the coercions between them handle the most common data types.
- Associative arrays use ordinary strings as the index in the array and offer an easy way to implement a single-key database.
- Regular expressions are a uniform notation for describing patterns of text.

OS/2 $199

- Default initialization and the absence of declarations shorten programs.

Large Model & Math Support

PolyAWK is a large model implementation and can use all of available memory to run big programs or read files greater than 64K. PolyAWK also includes extensive support for math functions such as strings, integers, floating point numbers and transcendental functions (sin, log, etc.) for scientific applications. Conversion between these types is automatic and always optimized for speed without compromising accuracy.

PolyAWK Comes With The Book

When you order PolyAWK you receive a copy of *The AWK Programming Language* written by the authors of the original UNIX-based AWK. The book begins with a tutorial that shows how easy AWK is to use, followed by a comprehensive manual. Because PolyAWK is a complete implementation of AWK as defined by the book's authors, you will use this book as the manual for PolyAWK.

PolyShell Bonus For MS-DOS!

PolyShell gives you 96 of the most useful UNIX commands and utilities under MS-DOS in less than 20K. You can still use MS-DOS commands at any time and exit or restart PolyShell without rebooting. MS-DOS programmers — discover what you have been missing! UNIX programmers — switch to MS-DOS painlessly! PolyShell and PolyAWK are each $99 when ordered separately. Save $50 by ordering the PolyShell + PolyAWK combination package for $149 (MS-DOS only).

30-Day Money Back Guarantee

You can purchase PolyAWK and the book, *The AWK Programming Language*, for $99 (MS-DOS) or $199 (OS/2), or $249 for both. If you already have the book, deduct $14. The combination package includes only one book. Order PolyAWK (MS-DOS & OS/2) plus PolyShell (MS-DOS) for $348.

To Order: 1-800-547-4000

Ask for Dept. PH. Send Checks and POs to POLYTRON Corporation, 1700 NW 167th Place, Beaverton, OR 97006, (503) 645-1150 — FAX: (503) 645-4576, TELEX: 325800 POLYTRON.

Call About Our OEM/VAR Strategic Alliance Program.

POLYTRON

High Quality Software Since 1982

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Saving Screens, Color Selection, and Other Mysteries

by
Michael Abrash

Installment 9: In which we answer questions about screen saving and color selection and look at ingenious reader solutions to yet another MASM mystery.

A couple of interesting reader questions arrived recently, as well as solutions to my latest MASM puzzle, and together they'll make for a varied and interesting column. First, we'll look at saving and restoring hi-res EGA and VGA screens. Next, we'll unravel the 16-out-of-64 colors mess. We'll wind up by discussing MASM locals within REPT blocks.

That's a lot of ground to cover, so let's get started!

Saving and Restoring EGA and VGA Screens

Frank Baker of the University of Wisconsin-Madison, wrote to ask how one goes about saving and restoring EGA hi-res (640 x 350) graphics screens to and from disk. Well, Frank, if you understood everything we've discussed in the first eight installments of the *On Graphics* series, you should be able to solve this one on your own . . . but I agree that it's not obvious, and it certainly is useful.

The basic principle for saving EGA and VGA hi-res graphics screens is astonishingly simple: Write each plane to disk separately. All we need do is enable reads from plane 0 and write the 28,000 bytes of plane 0 that are displayed in mode 10h to disk, then enable reads from plane 1 and write the displayed portion of that plane to disk, and so on for planes 2 and 3. The result is a file that's 112,000 (28,000 * 4) bytes long, with the planes stored as four distinct 28,000-byte blocks, as shown in Figure 1.

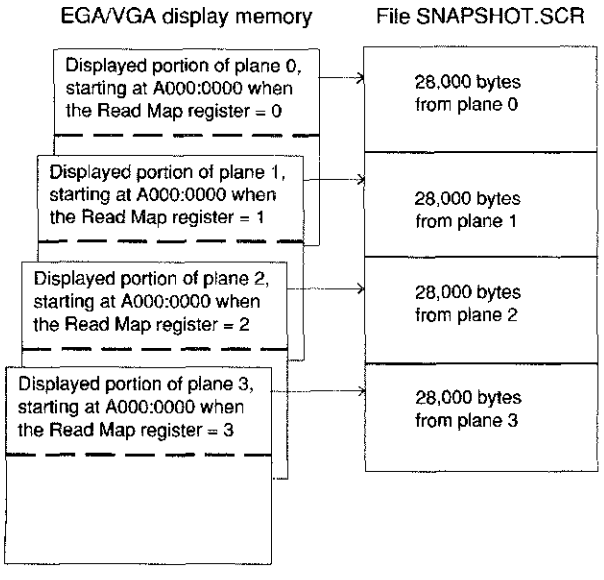

EGA/VGA display memory
File SNAPSHOT.SCR
Displayed portion of plane 0, starting at A000:0000 when the Read Map register = 0
Displayed portion of plane 1, starting at A000:0000 when the Read Map register = 1
Displayed portion of plane 2, starting at A000:0000 when the Read Map register = 2
Displayed portion of plane 3, starting at A000:0000 when the Read Map register = 3
28,000 bytes from plane 0
28,000 bytes from plane 1
28,000 bytes from plane 2
28,000 bytes from plane 3

Figure 1: EGA/VGA memory is saved by copying the 28,000 displayed bytes of each plane in turn to the file SNAPSHOT.SCR. The Read Map register is used to select which of the 4 planes is readable at A000:0000 at any given time.

The program shown in Listing 1 does just what I've described, putting the screen into mode 10h, putting up some bit-mapped text so that there's something to save, and creating the 112K file SNAPSHOT.SCR, which contains the visible portion of the hi-res bit map.

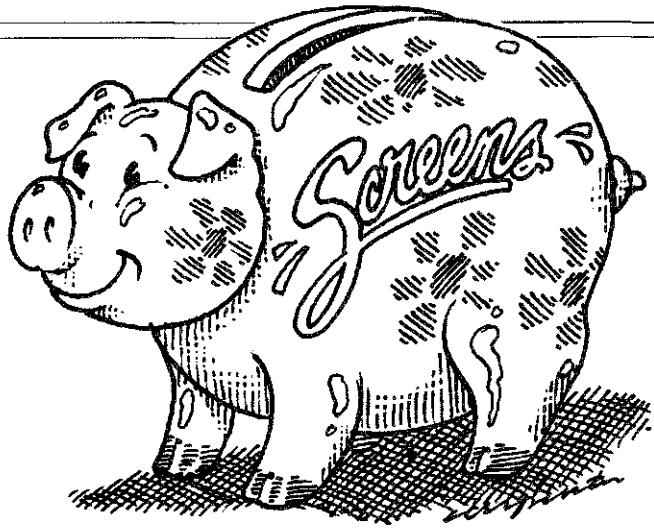
The only part of Listing 1 that's even remotely tricky is the use of the Read Map register (Graphics Controller register 4) to make each of the four planes of display memory readable in turn. The same code is used to write 28,000 bytes of display memory to disk four times, and 28,000 bytes of memory starting at A000:0000 are written to disk each time; however, a different plane is read each time, thanks to the changing setting of the Read Map register. (If this is unclear, refer back to Figure 1; you may also want to reread *On Graphics* #7 (*PJ* 6.4, July/August 1988) to brush up on the operation of the Read Map register in particular and reading EGA memory in general.)

Of course, we'll want the ability to restore what we've saved, and Listing 2 does this. Listing 2 reverses the action of Listing 1, selecting mode 10h and then loading 28,000 bytes from SNAPSHOT.SCR into each plane of display memory. The Map Mask register (Sequence Controller register 2) is used to select the plane to be written to. If your computer is slow enough, you can see the colors of the text change as each plane is loaded when Listing 2 runs. Note that Listing 2 does not itself draw any text, but rather simply loads the bit map saved by Listing 1 back into the hi-res bit map.

If you compare Listings 1 and 2, you will see that the Map Mask register setting used to load a given plane does not match the Read Map register setting used to read that plane. This is so because while only one plane can ever be read at a time, anywhere from zero to four planes can be written to at once; consequently, Read Map register settings are plane selections from 0 to 3 while Map Mask register settings are plane *masks* from 0 to 15, where a bit 0 setting of 1 enables writes to plane 0, a bit 1 setting of 1 enables writes to plane 1, and so on. Again, *On Graphics* #7 provides a detailed explanation of the differences between the Read Map and Map Mask registers.

Screen saving and restoring is pretty simple, eh? There are a few *caveats*, of course, but nothing serious. First, the EGA's registers must be programmed properly in order for screen saving and restoring to work. For screen saving, you must be in read mode 0; if you're in color compare mode, there's no

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



telling what bit pattern you'll save, but it certainly won't be the desired screen image. For screen restoring, you must be in write mode 0, with the Bit Mask register set to 0FFh and the Data Rotate register set to 0 (no data rotation and the logical function set to pass the data through unchanged).

While these requirements are no problem if you're simply calling a subroutine in order to save an image from your program, they pose a considerable problem if you're designing a hot-keyed TSR that can capture a screen image at any time. With the EGA, there's never any way to tell what state the registers are currently in since the registers aren't readable. As a result, any TSR that sets the Bit Mask to 0FFh, the Data Rotate register to 0, and so on runs the risk of interfering with the drawing code of the program that's already running.

What's the solution? Frankly, the solution is to get a VGA. A TSR designed for the VGA can simply read out and save the state of the registers of interest, program those registers as needed, save the screen image, and restore the original settings. From a programmer's perspective, readable registers are certainly near the top of the list of things to like about the VGA!

If you are going to write a hi-res VGA version of the screen capture program, be sure to account for the increased size of the VGA's mode 12h bit map. The mode 12h (640 x 480) screen uses 37.5K per plane of display memory, so for mode 12h the displayed screen size equate in Listings 1 and 2 should be changed to

```
<<<start of code>>>
DISPLAYED_SCREEN_SIZE    equ   (640/8)*480
<<<end of code>>>
```

Similarly, if you're capturing a graphics screen that starts at an offset other than 0 in the segment at A000, you must change the memory offset used by the disk functions to match. You can, if you so desire, read the start offset of the display memory providing the information shown on the screen from the Start Address registers (CRT Controller registers 0Ch and 0Dh).

Finally, be aware that the screen capture and restore programs in Listings 1 and 2 are appropriate only for EGA/VGA modes 0Dh, 0Eh, 0Fh, 010h, and 012h since they assume a four-plane configuration of EGA/VGA memory. In all text modes and in CGA graphics modes, and in VGA modes 11h and 13h as well, display memory can simply be written to disk and read back as a linear block of memory just like a normal array.

Listings 1 and 2 are written in assembler, but the principles they illustrate apply equally well to high-level languages. In fact, there's no need for any assembler at all when saving an EGA/VGA screen, so long as the high-level language you're using can perform direct port I/O to set up the adapter and can read and write display memory directly.

One tip if you're saving and restoring the screen from a high-level language on an EGA, though: After you've completed the save or restore operation, be sure to put any registers you've changed back to their default settings. Some high-level languages (and the BIOS as well) assume that various registers are left in a certain state, so on the EGA it's safest to leave the registers in their most likely state. On the VGA, of course, you can just read the registers out before you change them, then leave them as you found them when you're done.

16 Colors out of 64

Eric Klien, of Chelmsford, Massachusetts, wrote to ask how one produces the 64 colors from which the 16 colors displayed by the EGA can be chosen. The answer is simple enough: There's a BIOS function that lets you select the mapping of the 16 possible pixel values to the 64 possible colors. Let's establish a bit of background before proceeding, however.

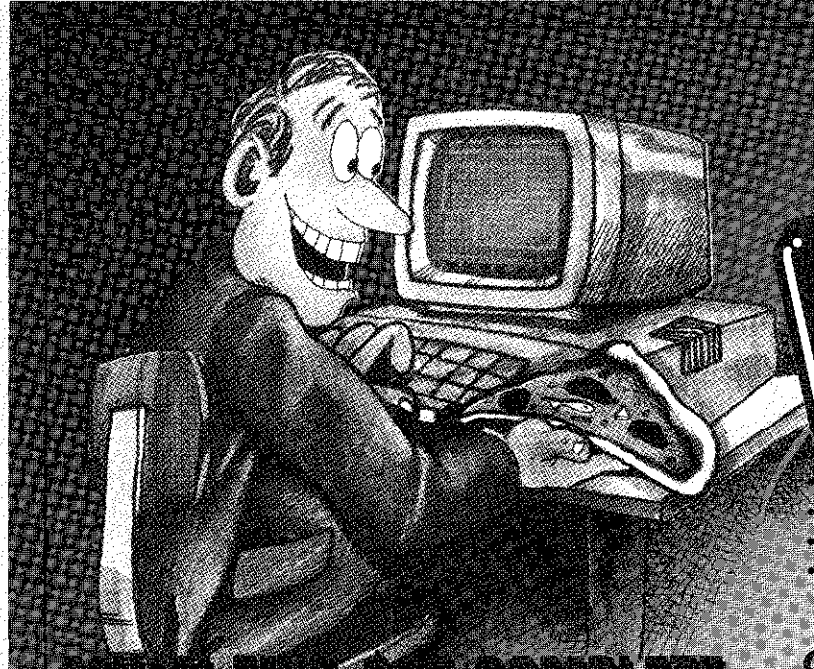
Frankly, the solution is to get a VGA.

The EGA sends pixel information to the monitor on 6 pins. This means that there are 2 to the 6th, or 64 possible colors that an EGA can generate. However, in 200-scan-line modes Enhanced Color Display-compatible monitors ignore two of the signals, so in CGA-compatible modes (modes 4, 5, 6, and the 200-scan-line versions of modes 0, 1, 2, and 3) you can select from only 16 colors (although the colors can still be remapped, as described below). If you're hooked up to a monitor that isn't capable of displaying 350 scan lines (such as the old IBM Color Display), you can never select from more than 16 colors since those monitors accept only four input signals. For now, we'll assume we're in one of the 350-scan-line color modes, a group that includes modes 0Dh, 0Eh, and 10h and the 350-scan-line versions of modes 0, 1, 2, and 3.

Each pixel comes out of memory (or, in text mode, out of the attribute-handling portion of the EGA) as a 4-bit value, denoting 1 of 16 possible colors. In graphics modes, the 4-bit pixel value is made up of one bit from each plane, with 8 pixels' worth of data stored at any given byte address in display memory. Normally, we think of the 4-bit value of a pixel as being that pixel's color, so a pixel value of 0 is black, a pixel value of 1 is blue, and so on, as if that's a built-in feature of the EGA.

Hah! The correspondence of pixel values to color is absolutely arbitrary, depending solely on how the color-mapping portion of the EGA containing the palette registers is programmed. If you cared to have color 0 be bright red and

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Multi-Edit vs. PIZZA

With EVERYTHING!

- Is your editor OUT TO LUNCH?
- Does it handle ALL OF YOUR NEEDS?
- Is it flexible, programmable and reconfigurable?
- MOST IMPORTANTLY, Is it EASY TO USE?

OR WOULD YOU RATHER BE EATING PIZZA?

MULTI-EDIT vs. the COMPLETE VERSION 2.0

	Multi-Edit	BRIEF 2.0	Norton Editor	Vedit Plus	PIZZA WITH EVERYTHING
Edit 20+ Files larger than memory	Yes	Yes	No	Yes	12 slices
Powerful high level macro language	Yes	Yes	No	Yes	Italian
Full UNDO	Yes	Yes	No	No	No
Visual marking of blocks	Yes	Yes	Yes	No	Looks Good
Line, stream and column blocks	Yes	Yes	No	No	Use Knife
Automatic file save	Yes	Yes	No	No	No
Online help	Extensive	Limited	Limited	Limited	Extensive
Choice of keystroke commands or menu system	Yes	No	No	Yes	Menu Available
Function Key assignments labeled on screen (may be disabled)	Yes	No	No	No	No
Word processing functions	Extensive	Limited	Limited	Extra Cost	Difficult
Complete DOS shell	Yes	No	No	No	Deep Dish
Pop-up Programmer's Calculator and ASCII Table	Yes	No	No	No ASCII	No
Unlimited Off the Cuff keystroke macros	Yes	No	No	Yes	Sauce on Cuff often
Allocates all available memory to compiler when run from within editor	Yes	No	No	No	Lots of bytes
Intelligent indenting, template editing and brace/parenthesis/block matching and checking for C, PASCAL, BASIC and MODULA-2	Yes	C Only	No	Limited	Limited Intelligence
Flexible condensed mode display	Yes	No	Yes	No	Definitely
PRICE	$99	$195	$90	$195	About $12

Multi-Edit is a powerful, easy-to-use programmer's editor with a full-featured word processor, a complete DOS shell, and a pop-up programmer's calculator and ASCII table. Multi-Edit is the only editor that automatically manages virtual memory for you, so you can edit multiple files regardless of their size. Multi-Edit's powerful macro language is easy to read and write, and comes with macros for C, PASCAL, BASIC and MODULA-2 that provide intelligent indenting, template editing and brace/parenthesis/block matching and checking. Multi-Edit's word processor has automatic word-wrap, pagination, full print formatting with justification, bold type, underlining and centering, flexible line drawing, and even a table of contents generator. Multi-Edit lets you compile within the editor, automatically positions the cursor at errors, and allocates all available memory to the compiler. Multi-Edit's DOS shell has a scrollable directory listing, lets you copy, delete and load multiple files with one command, and supports background file printing. Multi-Edit has regular expression search and translate, condensed mode display for easy viewing of your program structure, and a pop-up full-function programmer's calculator and ASCII chart.

And MOST IMPORTANT, Multi-Edit has the BEST USER-INTERFACE ON THE MARKET! Extensive context-sensitive help, choice of full menu system or logical function key layout, function keys are always labeled on screen (no guessing required!), keyboard may be easily reconfigured and re-labeled, extensive mouse support, easy, automatic recording and playback of keystrokes, anchovies easily removed.

MULTI-EDIT COMBINES POWER WITH EASE OF USE LIKE NO OTHER EDITOR ON THE MARKET TODAY.

Only MULTI-EDIT tastes this good!

Fully automatic Windowing and Virtual Memory
- Edit multiple files regardless of physical memory size
- Easy cut-and-past between files
- View different parts of the same file

Powerful, EASY-TO-READ high-level macro language
- Standard language syntax
- Full access to ALL Editor functions

Language-specific macros for C, PASCAL, BASIC and MODULA-2
- Smart Indenting
- Smart brace/parenthesis/block checking
- Template editing
- More languages on the way

Terrific word-processing features for all your documentation needs
- Intelligent word-wrap
- Automatic pagination
- Full print formatting with justification, bold type, underlining and centering
- Flexible line drawing
- Even a table of contents generator

Compile within the editor
- Automatically positions cursor at errors
- Allocates all available memory to compiler

Complete DOS Shell.
- Scrollable directory listing
- Copy, Delete and Load multiple files with one command
- Background file printing

Regular expression search and translate
Condensed Mode display, for easy viewing of your program structure
Pop-up FULL-FUNCTION Programmer's Calculator and ASCII chart

and MOST IMPORTANT, the BEST USER-INTERFACE ON THE MARKET!
- Extensive context-sensitive help
- Choice of full menu system or logical function key layout
- Function keys are always labeled on screen (no guessing required!)
- Keyboard may be easily reconfigured and re-labeled
- Extensive mouse support
- Easy, automatic recording and playback of keystrokes
- Anchovies easily removed

Get Our FULLY FUNCTIONAL DEMO Copy for only $10!

To Order, Call 24 hours a day:
1-800-221-9280 Ext. 951
In Arizona: 1-602-968-1945
Credit Card and COD orders accepted.

American Cybernetics
1228 N. Stadem Dr.
Tempe, AZ 85281

Requires IBM/PC/XT/AT/PS2 or full compatible, 256K RAM, PC/MS-DOS 2.0 or later. Multi-Edit and American Cybernetics are trademarks of American Cybernetics. BRIEF is a trademark of Underware, Inc. Norton Editor is a trademark of Peter Norton Computing, Inc. Vedit is a registered trademark of CompuView Products Inc. Copyright 1987 by American Cybernetics.

Number 184 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

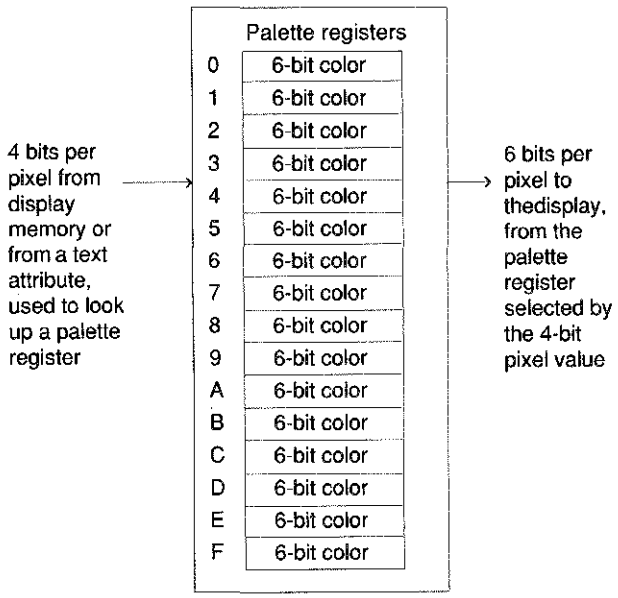

Palette registers
0 6-bit color
1 6-bit color
2 6-bit color
3 6-bit color
4 6-bit color
5 6-bit color
6 6-bit color
7 6-bit color
8 6-bit color
9 6-bit color
A 6-bit color
B 6-bit color
C 6-bit color
D 6-bit color
E 6-bit color
F 6-bit color
4 bits per pixel from display memory or from a text attribute, used to look up a palette register
6 bits per pixel to thedisplay, from the palette register selected by the 4-bit pixel value

Figure 2: The 16 palette registers translate each 4-bit pixel value from display memory or a text attribute into a 6-bit pixel color to send to the display.

color 1 be black, that could easily be arranged, as could a mapping in which all 16 colors were yellow. What's more, these mappings affect text-mode characters as readily as they do graphics-mode pixels, so you could map text attribute 0 to white and text attribute 15 to black to produce a Mac-like black on white display if you wished.

Each of the 16 palette registers stores the mapping of one of the 16 possible 4-bit pixel values from memory to one of 64 possible 6-bit pixel values to be sent to the monitor as video data, as shown in Figure 2. A 4-bit pixel value of 0 causes the 6-bit value stored in palette register 0 to be sent to the display as the color of that pixel, a pixel value of 1 causes the contents of palette register 1 to be sent to the display, and so on. Since there are only four input bits, it stands to reason that only 16 colors are available at any one time; since there are six output bits, however, those 16 colors can be mapped to any of 64 colors. The mapping for each of the 16 pixel values is controlled by the lower six bits of the corresponding palette register, as shown in Figure 3. Secondary red, green, and blue are less-intense versions of red, green, and blue although their exact effects vary from monitor to monitor. The best way to figure out what the 64 colors look like on your monitor is to see them, and that's just what the program in Listing 3, which we'll discuss shortly, lets you do.

How are the palette registers set? Well, it's certainly possible to set the palette registers directly, addressed at registers 0 through 0Fh of the Attribute Controller. However, setting the palette registers is a bit tricky—bit 5 of the Attribute Controller Index register must be 0 while the palette registers are written to, and glitches can occur if the updating doesn't take place during the blanking interval—and besides, it turns out that there's no need at all to go straight to the hardware on this one. Conveniently, the EGA BIOS provides us with video function 10h, which supports setting either any one palette register or all 16 palette registers (and the overscan register as well) with a single video interrupt.

Video function 10h is invoked by performing an INT 10h with AH set to 10h. If AL is 0 (subfunction 0), BL contains the number of the palette register to set and BH contains the value to set that register to. If AL is 1 (subfunction 1), BH contains the value to set the overscan register to. Finally, if AL is 2 (subfunction 2), ES:DX points to a 17-byte array containing the values to set palette registers 0–15 and the overscan register to. (For completeness, although it's unrelated to the palette registers, there is one more subfunction of video function 10h. If AL = 3 [subfunction 3], bit 0 of BL is set to 1 to cause bit 7 of text attributes to select blinking, or set to 0 to cause bit 7 of text attributes to select high-intensity reverse video.)

Listing 3 uses video function 10h, subfunction 2 to step through all 64 possible colors. This is accomplished by putting up 16 color bars, one for each of the 16 possible 4-bit pixel values, then changing the mapping provided by the palette registers to select a different group of 16 colors from the set of 64 each time a key is pressed. Initially, colors 0–15 are displayed, then 1–16, then 2–17, and so on up to color 3Fh wrapping around to colors 0–14, and finally back to colors 0–15. (By the way, at mode set the 16 palette registers are not set to colors 0–15, but rather to 0h, 1h, 2h, 3h, 4h, 5h, 14h, 7h, 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, and 3Fh, respectively. Bits 6, 5, and 4—secondary red, green, and blue—are all set to 1 in palette registers 8–15 in order to produce high-intensity colors. Palette register 6 is set to 14h to produce brown, rather than the yellow that the expected value of 6h would produce.)

When you run Listing 3, you'll see that the whole screen changes color as each new color set is selected. This occurs because most of the pixels on the screen have a value of 0, selecting the background color stored in palette register 0, and we're reprogramming palette register 0 right along with the other 15 palette registers.

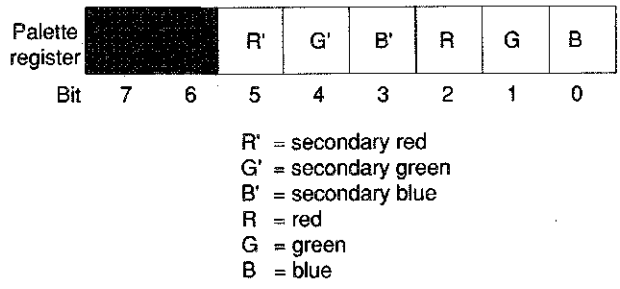


Figure 3: The organization of color bits within a palette register.

It's important to understand that in Listing 3 the contents of display memory are never changed after initialization. The only change is the mapping from the 4-bit pixel data coming out of display memory to the 6-bit data going to the monitor. For this reason, it's technically inaccurate to speak of bits in display memory as representing colors; more accurately, they represent attributes in the range 0–15, which are mapped to colors 0–3Fh by the palette registers.

While we're at it, I'm going to touch on overscan. Overscan is the color of the border of the display, the rectangular area around the edge of the monitor that's outside the region displaying active video data but inside the blanking area. The overscan (or border) color can be programmed to any of the 64 possible colors by either setting Attribute Controller register 12h directly or calling video function 10h, subfunction 1. On ECD-compatible monitors, however, there's too

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

little scan time to display a proper border when the EGA is in 350-scan-line mode, so overscan should always be 0 (black) unless you're in 200-scan-line mode. Note, though, that a VGA can easily display a border on a VGA-compatible monitor, and VGAs are in fact programmed at mode set for an 8-pixel-wide border in all modes; all you need do is set the overscan color on any VGA to see the border.

An interesting point: The Attribute Controller provides a very convenient way to blank the screen, in the form of the aforementioned bit 5 of the Attribute Controller Index register (at address 3C0h after the Input Status register 1 [3DAh in color, 3BAh in monochrome] has been read and on every other write to 3C0h thereafter). Whenever bit 5 of the AC Index register is 0, video data is cut off, effectively blanking the screen. Setting bit 5 of the AC Index back to 1 restores video data immediately. Listing 4 illustrates this simple but effective form of screen blanking.

Does that do it for color selection? Yes and no. For the EGA, we've covered the whole of color selection—but not so for the VGA. The VGA can emulate everything we've discussed, but actually performs one 4-bit to 8-bit translation, followed by yet another translation, this one 8-bit to 18-bit. What's more, the VGA has the ability to flip instantly through as many as sixteen 16-color sets. The VGA's color selection capabilities, which are supported by another set of BIOS functions, can be used to produce stunning color effects. Then, too, there are modes 11h and 13h of the VGA, which don't fit neatly into the 16-color scheme we've been discussing. However, I'm running out of room, and since everything we've discussed applies equally well to the EGA and VGA, we've covered the most broadly useful information about color selection; I'll try to get to VGA-specific color selection soon.

MASM Stuff: Readers to the Rescue

A few months back I pointed out that although the MASM manual indicated otherwise, the LOCAL directive didn't work properly inside a REPT block, and I asked if any *PJ* readers had workarounds. As usual, several readers did; all the solutions are good, and one is a wonderful illustration of the lengths to which people are willing to go in order to get MASM to cooperate.

First the simple solution. Both Michael Liebert of Custom Real-Time Software in Caldwell, New Jersey, and Bradford Levy of BKL Consulting in Lawrence, Kansas, sent in the solution shown in Listing 5. (I've reproduced Brad's code since it was in source form rather than assembler listing format, but the two solutions are essentially the same.) The solution is reasonably simple, although requiring two nested macros in order to get local labels into REPT blocks seems a bit much. As Brad commented, "There are a number of things that you would think that MASM 5.0 should let you do more directly." Indeed there are.

The third solution, submitted by Adrian Crum of La Habra, California, is shown in Listing 6. Adrian has come up with the ingenious solution of creating and maintaining *his own temporary label names*! This is exactly what MASM is supposed to do with local labels inside REPT blocks, of course; but since MASM doesn't do it, Adrian simply did it himself. It's not as convenient as it would be if LOCAL worked properly in REPT blocks—the local label counter ?CNT1 has to be explicitly initialized and incremented—but it's certainly a

SCREEN MANAGER
For C, BASIC, FORTRAN, COBOL Programmers

The Screen Generator ™

The Screen Generator offers professional screen management for creation of Window, Help, Menu, and Data Entry Screens for over 20 compilers. You create screens using SG's Screen Painter. All screen text and field information is stored in compact screen libraries. All Screen Display and Data Entry functions are controlled from within your program. SG's function calls provide all the flexibility and speed you'll ever need for screen display and block mode or field level data entry. The Screen Generator includes: The Screen Painter/Print Utility, The Screen Manager, and 100 page User Guide.

So give us a call! You'll be up and running in no time, while our 60-day Money Back Guarantee eliminates all the risk!

The Screen Generator ™ — New Version

C, BASIC, ASM & Turbo Pascal	$79.
FORTRAN or COBOL	add $20.
Demo Disk Available	VISA/MC

The West Chester Group
P.O. Box 1304, West Chester, PA 19380
(215) 644-4206

Number **196** on the Reader Service Card

EASY FAST PROFESSIONAL

DATA ENTRY
WINDOWS
MENUS
HELP

If you are serious about programming
PLEASE try HI-SCREEN XL!

HI-SCREEN XL™

$149 only

Multilanguage support
No Royalties
30-day risk free

Call now for demo and information:
1-800-338-2852
in CA: (415) 397-4666

"You may like other screen management tools, but you will love HI-SCREEN XL."

Softway, Inc., 500 Sutter St., Suite 222, San Francisco, CA 94102

Number **110** on the Reader Service Card

PRODUCED 2008 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

creative use of MASM's macro capabilities . . . and it does work. Listing 6 is an excellent tutorial on some advanced aspects of macro usage; if you're interested in making good use of macros, I suggest you study this listing until you understand every bit of it.

In Conclusion

Gosh. We certainly get into interesting areas when you folks come up with the questions and answers. I've cleared the queue of pending reader feedback, so at the moment I'm not sure what I'll look at next; as always, I welcome your suggestions. At any rate, rest assured that many unexplored mysteries yet lurk within the EGA and VGA, and we'll look into at least one of those next time.

See you then. ♦

Michael Abrash is a Senior Software Engineer for Orion Instruments, Inc., of Redwood City, California, where he is working on the design of advanced PC-based test instrumentation. Readers may write to Michael at P.O. Box 39051, Mountain View, CA 94039, or drop Michael a line via MCI Mail (user name MABRASH).

Code follows.

The Electronic Abacus

An abacus uses a number system that might be called bi-quinary. A row of beads above a horizontal bar (the "binary" part) designates whether the four rows of beads below the bar (the "quinary" part) are to be interpreted as 0–4 or 5–9. The IBM 650 of the late 1950s used the same number system. It stored decimal digits, each digit represented by seven bits, with exactly two on. Two bits represented the range, one representing 0–4 and the other 5–9. The remaining five bits represented the digits 0–4 or 5–9, respectively, depending upon which range bit was on.

The total storage capacity of the 650 on which I cut my computing teeth was 1000 10-digit words. The storage medium was a rotating drum. And that was that! One Wednesday a notice came around cautioning us not to schedule any computer time for the following weekend. IBM was coming in to install their new super-gee-whiz 2000-word drum.

We sat around for the next two weeks wondering what the hell we were going to do with all that extra memory!

And the whole thing needed 15 tons of air conditioning!—*W.G. Madison* ♦

Unbelievable!

SOURCER™

■ SEE HOW PROGRAMS WORK
■ EASILY MODIFY PROGRAMS

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Includes a definition file facility to include your own remarks and descriptive labels, force data types, and more. Complete support for 8088/87 through 80286/287 and V20/V30 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

On my list of programs that I simply won't do without!
—Robert Hummel, Senior Technical Editor, PC Magazine

SAMPLE OUTPUT

Fully automatic
Program header
Assembler directives
Determines data areas and type
Detailed comments
Simulator follows segment changes
Easy to read format

```
resetprn.lst    ResetPRN v1.01              Sourcer Listing    19-Apr-88  4:05 pm      Page 1

                                PAGE  60,132
                                ;
                                ;
                                ;                    RESETPRN
                                ;
                                ;       Created:   15-Apr-88
                                ;       Version:   1.01
                                ;
                                ;

= 0008                          data_1e         equ     8               ; (0040:0008=378h)

                                ;-------------------------------------------------------- seg_a ----
                                seg_a           segment para public
                                                assume cs:seg_a, ds:seg_a, ss:stack_seg_b

                                resetprn        proc    far
658E:0000                       start:
658E:0000  EB 23                                jmp     short loc_1
658E:0002  52 65 73 65 74 50                    db      'ResetPRN v1.01', 0Dh
658E:0008  52 4E 20 76 31 2E
658E:000E  30 31 0D
658E:0011  0040                 data_2          dw      40h
658E:0013  0D 0A 52 65 73 65    data_3          db      0Dh, 0Ah, 'Reset Printer? $'
658E:0019  74 20 50 72 69 6E
658E:001F  74 65 72 3F 20 24
658E:0025                       loc_1:
658E:0025  0E                                   push    cs
658E:0026  1F                                   pop     ds
658E:0027  BA 0013                              mov     dx,offset data_3  ; (658E:0013=0Dh)
658E:002A  B4 09                                mov     ah,9
658E:002C  CD 21                                int     21h               ; DOS Services  ah=function 09h
                                                                          ;  display char string at ds:dx
658E:002E  B4 01                                mov     ah,1
658E:0030  CD 21                                int     21h               ; DOS Services  ah=function 01h
                                                                          ;  get keybd char al, with echo
658E:0032  3C 79                                cmp     al,79h            ; 'y'
658E:0034  75 16                                jne     loc_3             ; Jump if not equal
658E:0036  8E 1E 0011                           mov     ds,data_2         ; (658E:0011=40h)
658E:003A  8B 16 0008                           mov     dx,ds:data_1e     ; (0040:0008=378h)
658E:003E  83 C2 02                             add     dx,2
658E:0041  B0 08                                mov     al,8
658E:0043  EE                                   out     dx,al             ; port 37Ah, printer-2 control
                                                                          ;  al = 8, initialize printer
658E:0044  B9 8000                              mov     cx,8000h
658E:0047                       locloop_2:
658E:0047  E2 FE                                loop    locloop_2         ; Loop if cx > 0
658E:0049  B0 0C                                mov     al,0Ch
658E:004B  EE                                   out     dx,al             ; port 37Ah, printer-2 control
                                                                          ;  al = 0Ch, init & strobe off
658E:004C                       loc_3:
658E:004C  B4 4C                                mov     ah,4Ch            ; 'L'
658E:004E  CD 21                                int     21h               ; DOS Services  ah=function 4Ch
                                                                          ;  terminate with al=return code
                                resetprn        endp
                                seg_a           ends

                                ;-------------------------------------------------------- stack_seg_b ----
                                stack_seg_b     segment para stack
6693:0000  00C0[FF]                             db      192 dup (0FFh)
                                stack_seg_b     ends

                                                end     start
```

(Source code output and inline cross reference can also be selected)

BIOS SOURCE

■ CHANGE AND ADD FEATURES
■ CLARIFY INTERFACES

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER $99.95 BIOS Pre-Processor* $49.95 SOURCER w/BIOS Pre-Processor $139.95

USA Shipping & Handling $3; Outside USA $15; CA Residents add local sales tax 6, 6.5 or 7%; *requires SOURCER

All our products come with a 30 day money back satisfaction guarantee. Not copy protected. To order or receive additional information just call!

VISA **(800) 662-8266** MasterCard

V COMMUNICATIONS
3031 Tisch Way, Suite 905, Dept. J3, San Jose, CA 95128 (408) 296-4224

PS/2, AT, XT, and PC are trademarks of IBM Corp.

Number 125 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk.
ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386.
Mail order information on pages 8 and 88.

Code from "Saving VGA Screens" by Michael Abrash

1

```
;
; *** Listing 1 ***
;
; Program to put up a mode 10h EGA graphics screen, then save it
; to the file SNAPSHOT.SCR.
;
VGA_SEGMENT     equ     0a000h
GC_INDEX        equ     3ceh    ;Graphics Controller Index register
READ_MAP        equ     4       ;Read Map register index in GC
DISPLAYED_SCREEN_SIZE equ (640/8)*350
                                ;# of displayed bytes per plane in a
                                ; hi-res graphics screen
;
stack   segment para stack 'STACK'
        db      512 dup (?)
stack   ends
;
Data    segment word 'DATA'
SampleText db   'This is bit-mapped text, drawn in hi-res '
        db      'EGA graphics mode 10h.', 0dh, 0ah, 0ah
        db      'Saving the screen (including this text)...'
        db      0dh, 0ah, '$'
Filename db     'SNAPSHOT.SCR',0        ;name of file we're saving to
ErrMsg1 db      '*** Couldn''t open SNAPSHOT.SCR ***',0dh,0ah,'$'
ErrMsg2 db      '*** Error writing to SNAPSHOT.SCR ***',0dh,0ah,'$'
WaitKeyMsg db   0dh, 0ah, 'Done. Press any key to end...',0dh,0ah,'$'
Handle  dw      ?       ;handle of file we're saving to
Plane   db      ?       ;plane being read
Data    ends
;
Code    segment
        assume  cs:Code, ds:Data
Start   proc    near
        mov     ax,Data
        mov     ds,ax
;
; Go to hi-res graphics mode.
;
        mov     ax,10h  ;AH = 0 means mode set, AL = 10h selects
                        ; hi-res graphics mode
        int     10h     ;BIOS video interrupt
;
; Put up some text, so the screen isn't empty.
;
        mov     ah,9    ;DOS print string function
        mov     dx,offset SampleText
        int     21h
;
; Delete SNAPSHOT.SCR if it exists.
;
        mov     ah,41h  ;DOS unlink file function
        mov     dx,offset Filename
        int     21h
;
; Create the file SNAPSHOT.SCR.
;
        mov     ah,3ch  ;DOS create file function
        mov     dx,offset Filename
        sub     cx,cx           ;make it a normal file
        int     21h
        mov     [Handle],ax     ;save the handle
        jnc     SaveTheScreen   ;we're ready to save if no error
        mov     ah,9            ;DOS print string function
        mov     dx,offset ErrMsg1
        int     21h             ;notify of the error
        jmp     short Done      ;and done
;
; Loop through the 4 planes, making each readable in turn and
; writing it to disk. Note that all 4 planes are readable at
; A000:0000; the Read Map register selects which plane is readable
; at any one time.
```

2

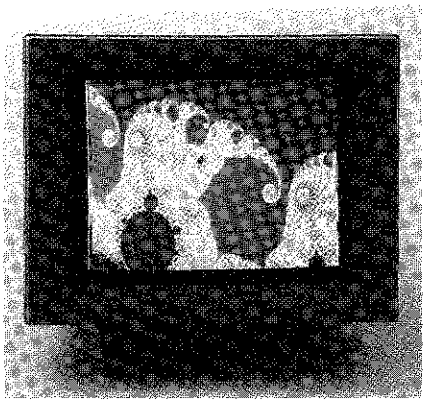
```
;
SaveTheScreen:
        mov     [Plane],0       ;start with plane 0
SaveLoop:
        mov     dx,GC_INDEX
        mov     al,READ_MAP     ;set GC Index to Read Map register
        out     dx,al
        inc     dx
        mov     al,[Plane]      ;get the # of the plane we want
                                ; to save
        out     dx,al   ;set to read from the desired plane
        mov     ah,40h  ;DOS write to file function
        mov     bx,[Handle]
        mov     cx,DISPLAYED_SCREEN_SIZE ;# of bytes to save
        sub     dx,dx   ;write all displayed bytes at A000:0000
        push    ds
        mov     si,VGA_SEGMENT
        mov     ds,si
        int     21h     ;write the displayed portion of this plane
        pop     ds
        cmp     ax,DISPLAYED_SCREEN_SIZE ;did all bytes get written?
        jz      SaveLoopBottom
        mov     ah,9            ;DOS print string function
        mov     dx,offset ErrMsg2
        int     21h             ;notify about the error
        jmp     short DoClose   ;and done
SaveLoopBottom:
        mov     al,[Plane]
        inc     ax      ;point to the next plane
        mov     [Plane],al
        cmp     al,3    ;have we done all planes?
        jbe     SaveLoop ;no, so do the next plane
;
; Close SNAPSHOT.SCR.
;
DoClose:
        mov     ah,3eh  ;DOS close file function
        mov     bx,[Handle]
        int     21h
;
; Wait for a keypress.
;
        mov     ah,9    ;DOS print string function
        mov     dx,offset WaitKeyMsg
        int     21h     ;prompt
        mov     ah,8    ;DOS input without echo function
        int     21h
;
; Restore text mode.
;
        mov     ax,3
        int     10h
;
; Done.
;
Done:
        mov     ah,4ch  ;DOS terminate function
        int     21h
Start   endp
Code    ends
        end     Start


;
; *** Listing 2 ***
;
; Program to restore a mode 10h EGA graphics screen from
; the file SNAPSHOT.SCR.
;
VGA_SEGMENT     equ     0a000h
SC_INDEX        equ     3c4h    ;Sequence Controller Index register
MAP_MASK        equ     2       ;Map Mask register index in SC
DISPLAYED_SCREEN_SIZE equ (640/8)*350
                                ;# of displayed bytes per plane in a
                                ; hi-res graphics screen
;
stack   se‡¥ment para stack 'STACK'
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

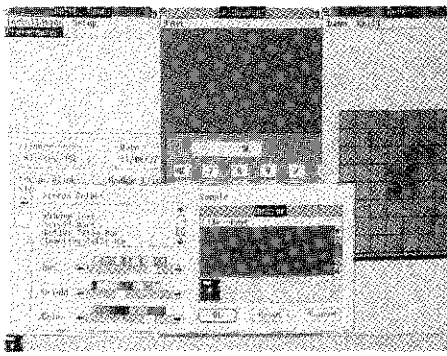
VERY HIGH RESOLUTION

The PC Tech COLOR and MONOCHROME video processor boards employ the TMS 3401 high performance graphics co-processor to insure the best possible video performanc at reasonable prices.



Color 34010 Video Processor:

- **Featured on the cover of Micro Cornucopia.**
- **From 800 x 512 through 1024 x 800 resolution (depending on monitor and configuration).**
- **8 Bits per pixel for 256 simultaneous colors**
- **Hardware support for CGA/MDA emulation.**
- **PC, XT, and AT compatible**

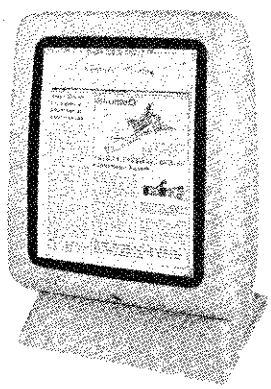


The PC Tech Color 34010 video processor is a superior 34010 native code and DGIS development tool. We support up to 4 megabytes of program (non-display) 34010 RAM as well as up to 768K bytes of display RAM. ***Compare our architecture and prices to any other intelligent graphics board. Then choose the PC Tech Color 34010 Video Processor for your development engine and your production requirements as well.***

Color 34010 Video Processor $1,350.00

Price includes 512K display RAM, 1024K program RAM, and utility software. Monitor not included.
Also available: DGIS, 34010 C compiler, assembler, 34010 fractal software,
additional display and program memory, and various monitor options.

PC Tech Monochrome 34010 Video Processor and Monitor



- **736 x 1024 resolution (other options available)**
- **2 bits per pixel for 4 hardware gray shades**
- **Hardware support for CGA/MDA/Hercules emulation**
- **PC, XT, and AT compatible**
- **Full page 66 line text editing with many popular editors**
- **Excellent windows 2.0 application development system**

The graphics and bit manipulation capabilities of the TMS 34010 make the PC Tech Monochrome 34010 Video Processor 66 line full page text and graphics display faster than many 25 line systems. The video processor is available separately or with the high resolution white phosphor monitor shown above.

Monochrome 34010 Video Sub-System $1,495.00

Price includes Monochrome Video Processor and monitor pictured above.
Also available: DGIS, TI 34010 C compiler, TI assembler.
Monochrome 34010 Video Processor also available separately.

Designed, Sold and Serviced By:

PTech

907 N. 6th St.
Lake City, MN 55041
(612) 345-4555
(612) 345-5514 (FAX)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash Code, Continued

3

```
        db      512 dup (?)
stack   ends
;
Data    segment word 'DATA'
Filename db     'SNAPSHOT.SCR',0 ;name of file we're restoring from
ErrMsg1 db      '*** Couldn't open SNAPSHOT.SCR ***',0dh,0ah,'$'
ErrMsg2 db      '*** Error reading from SNAPSHOT.SCR ***',0dh,0ah,'$'
WaitKeyMsg db   0dh, 0ah, 'Done. Press any key to end...',0dh,0ah,'$'
Handle  dw      ?       ;handle of file we're restoring from
Plane   db      ?       ;plane being written
Data    ends
;
Code    segment
        assume  cs:Code, ds:Data
Start   proc    near
        mov     ax,Data
        mov     ds,ax
;
; Go to hi-res graphics mode.
        int     10h
        mov     ax,10h  ;AH = 0 means mode set, AL = 10h selects
                        ; hi-res graphics mode
        int     10h     ;BIOS video interrupt
;
; Open SNAPSHOT.SCR.
;
        mov     ah,3dh  ;DOS open file function
        mov     dx,offset Filename
        sub     al,al           ;open for reading
        int     21h
        mov     [Handle],ax     ;save the handle
        jnc     RestoreTheScreen ;we're ready to restore if no error
        mov     ah,9            ;DOS print string function
        mov     dx,offset ErrMsg1
        int     21h             ;notify of the error
        jmp     short Done      ;and done
;
; Loop through the 4 planes, making each writable in turn and
; reading it from disk. Note that all 4 planes are writable at
; A000:0000; the Map Mask register selects which planes are readable
; at any one time. We only make one plane readable at a time.
;
RestoreTheScreen:
        mov     [Plane],0       ;start with plane 0
RestoreLoop:
        mov     dx,SC_INDEX
        mov     al,MAP_MASK     ;set SC Index to Map Mask register
        out     dx,al
        inc     dx
        mov     cl,[Plane]      ;get the # of the plane we want
                                ; to restore
        mov     al,1
        shl     al,cl   ;set the bit enabling writes to
                        ; only the one desired plane
        out     dx,al   ;set to read from the desired plane
        mov     ah,3fh  ;DOS read from file function
        mov     bx,[Handle]
        mov     cx,DISPLAYED_SCREEN_SIZE ;# of bytes to read
        sub     dx,dx   ;start loading bytes at A000:0000
        push    ds
        mov     si,VGA_SEGMENT
        mov     ds,si
        int     21h     ;read the displayed portion of this plane
        pop     ds
        jc      ReadError
        cmp     ax,DISPLAYED_SCREEN_SIZE ;did all bytes get read?
        jz      RestoreLoopBottom
ReadError:
        mov     ah,9            ;DOS print string function
        mov     dx,offset ErrMsg2
        int     21h             ;notify about the error
        jmp     short DoClose   ;and done
RestoreLoopBottom:
        mov     al,[Plane]
        inc     ax      ;point to the next plane
        mov     [Plane],al
        cmp     al,3    ;have we done all planes?
```

4

```
        jbe     RestoreLoop ;no, so do the next plane
;
; Close SNAPSHOT.SCR.
;
DoClose:
        mov     ah,3eh  ;DOS close file function
        mov     bx,[Handle]
        int     21h
;
; Wait for a keypress.
;
        mov     ah,8    ;DOS input without echo function
        int     21h
;
; Restore text mode.
;
        mov     ax,3
        int     10h
;
; Done.
;
Done:
        mov     ah,4ch  ;DOS terminate function
        int     21h
Start   endp
Code    ends
        end     Start


;
; *** Listing 3 ***
;
; Program to illustrate the color mapping capabilities of the
; EGA's palette registers.
;
VGA_SEGMENT     equ     0a000h
SC_INDEX        equ     3c4h    ;Sequence Controller Index register
MAP_MASK        equ     2       ;Map Mask register index in SC
BAR_HEIGHT      equ     14      ;height of each bar
TOP_BAR         equ     BAR_HEIGHT*6    ;start the bars down a bit to
                                        ; leave room for text
;
stack   segment para stack 'STACK'
        db      512 dup (?)
stack   ends
;
Data    segment word 'DATA'
KeyMsg  db      'Press any key to see the next color set. '
        db      'There are 64 color sets in all.'
        db      0dh, 0ah, 0ah, 0ah, 0ah
        db      13 dup (' '), 'Attribute'
        db      38 dup (' '), 'Color$'
;
; Used to label the attributes of the color bars.
;
AttributeNumbers        label   byte
x=      0
        rept    16
if x lt 10
        db      '0', x+'0', 'h', 0ah, 8, 8, 8
else
        db      '0', x+'A'-10, 'h', 0ah, 8, 8, 8
endif
x=      x+1
        endm
        db      '$'
;
; Used to label the colors of the color bars. (Color values are
; filled in on the fly.)
;
ColorNumbers    label   byte
        rept    16
        db      '000h', 0ah, 8, 8, 8, 8
        endm
COLOR_ENTRY_LENGTH      equ     ($-ColorNumbers)/16
        db      '$'
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

5

```
CurrentColor    db      ?
;
; Space for the array of 16 colors we'll pass to the BIOS, plus
; an overscan setting of black.
;
ColorTable      db      16 dup (?), 0
Data    ends
;
Code    segment
        assume  cs:Code, ds:Data
Start   proc    near
        cld
        mov     ax,Data
        mov     ds,ax
;
; Go to hi-res graphics mode.
;
        mov     ax,10h  ;AH = 0 means mode set, AL = 10h selects
                        ; hi-res graphics mode
        int     10h     ;BIOS video interrupt
;
; Put up relevant text.
;
        mov     ah,9    ;DOS print string function
        mov     dx,offset KeyMsg
        int     21h
;
; Put up the color bars, one in each of the 16 possible pixel values
; (which we'll call attributes).
;
        mov     cx,16   ;we'll put up 16 color bars
        sub     al,al   ;start with attribute 0
BarLoop:
        push    ax
        push    cx
        call    Bar_Up
        pop     cx
        pop     ax
        inc     ax      ;select the next attribute
        loop    BarLoop
;
; Put up the attribute labels.
;
        mov     ah,2    ;video interrupt set cursor position function
        sub     bh,bh   ;page 0
        mov     dh,TOP_BAR/14   ;counting in character rows, match to
                                ; top of first bar, counting in
                                ; scan lines
        mov     dl,16   ;just to left of bars
        int     10h
        mov     ah,9    ;DOS print string function
        mov     dx,offset AttributeNumbers
        int     21h
;
; Loop through the color set, one new setting per keypress.
;
        mov     [CurrentColor],0        ;start with color zero
ColorLoop:
;
; Set the palette registers to the current color set, consisting
; of the current color mapped to attribute 0, current color + 1
; mapped to attribute 1, and so on.
;
        mov     ah,(CurrentColor)
        mov     bx,offset ColorTable
        mov     cx,16   ;we have 16 colors to set
PaletteSetLoop:
        and     al,3fh          ;limit to 6-bit color values
        mov     [bx],al ;built the 16-color table used for setting
        inc     bx      ; the palette registers
        inc     ax
        loop    PaletteSetLoop
        mov     ah,10h  ;video interrupt palette function
        mov     al,2    ;subfunction to set all 16 palette registers
                        ; and overscan at once
        mov     dx,offset ColorTable
        push    ds
```

Code continues . . .

U.S. Postal Service

STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION

Required by 39 U.S.C. 3685)

1A. TITLE OF PUBLICATION	1B. PUBLICATION NO.	2. DATE OF FILING
Programmer's Journal	0 7 4 7 5 8 6 1	10/1/1988
3. FREQUENCY OF ISSUE: bi-monthly	3A. NO. OF ISSUES PUBLISHED ANNUALLY: 6	3B. ANNUAL SUBSCRIPTION PRICE: $24.00

4. COMPLETE MAILING ADDRESS OF KNOWN OFFICE OF PUBLICATION *(Street, City, County, State and ZIP+4 Code) (Not printers)*
150 North 4th Street, Springfield, OR 97477

5. COMPLETE MAILING ADDRESS OF THE HEADQUARTERS OF GENERAL BUSINESS OFFICES OF THE PUBLISHER *(Not printer)*
150 North 4th Street, Springfield, OR 97477

6. FULL NAMES AND COMPLETE MAILING ADDRESS OF PUBLISHER, EDITOR, AND MANAGING EDITOR *(This item MUST NOT be blank)*

PUBLISHER *(Name and Complete Mailing Address)*
Elizabeth Oakley, 150 North 4th Street, Springfield, OR 97477

EDITOR *(Name and Complete Mailing Address)*
Erin O'Connor, 150 North 4th Street, Springfield, OR 97477

MANAGING EDITOR *(Name and Complete Mailing Address)*

7. OWNER *(If owned by a corporation, its name and address must be stated and also immediately thereunder the names and addresses of stockholders owning or holding 1 percent or more of total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address, as well as that of each individual must be given. If the publication is published by a nonprofit organization, its name and address must be stated.) (Item must be completed.)*

FULL NAME	COMPLETE MAILING ADDRESS
Oakley Publishing Co.	150 N. 4th St., Springfield, OR 97477
Elizabeth Oakley	150 N. 4th St., Springfield, OR 97477
James Oakley	150 N. 4th St., Springfield, OR 97477

8. KNOWN BONDHOLDERS, MORTGAGEES, AND OTHER SECURITY HOLDERS OWNING OR HOLDING 1 PERCENT OR MORE OF TOTAL AMOUNT OF BONDS, MORTGAGES OR OTHER SECURITIES *(If there are none, so state)*

FULL NAME	COMPLETE MAILING ADDRESS
none	

9. FOR COMPLETION BY NONPROFIT ORGANIZATIONS AUTHORIZED TO MAIL AT SPECIAL RATES *(Section 423.12 DMM only)*
The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes *(Check one)*

(1) ☐ HAS NOT CHANGED DURING PRECEDING 12 MONTHS
(2) ☐ HAS CHANGED DURING PRECEDING 12 MONTHS *(If changed, publisher must submit explanation of change with this statement.)*

10. EXTENT AND NATURE OF CIRCULATION *(See instructions on reverse side)*	AVERAGE NO. COPIES EACH ISSUE DURING PRECEDING 12 MONTHS	ACTUAL NO. COPIES OF SINGLE ISSUE PUBLISHED NEAREST TO FILING DATE
A. TOTAL NO. COPIES *(Net Press Run)*	19,388	23,591
B. PAID AND/OR REQUESTED CIRCULATION 1. Sales through dealers and carriers, street vendors and counter sales	6,407	12,033
2. Mail Subscription *(Paid and/or requested)*	8,087	8,218
C. TOTAL PAID AND/OR REQUESTED CIRCULATION *(Sum of 10B1 and 10B2)*	14,494	20,251
D. FREE DISTRIBUTION BY MAIL, CARRIER OR OTHER MEANS SAMPLES, COMPLIMENTARY, AND OTHER FREE COPIES	1,269	3,113
E. TOTAL DISTRIBUTION *(Sum of C and D)*	15,763	23,364
F. COPIES NOT DISTRIBUTED 1. Office use, left over, unaccounted, spoiled after printing	578	219
2. Return from News Agents	3,047	8
G. TOTAL *(Sum of E, F1 and 2--should equal net press run shown in A)*	19,388	23,591

11. I certify that the statements made by me above are correct and complete

SIGNATURE AND TITLE OF EDITOR, PUBLISHER, BUSINESS MANAGER, OR OWNER: Elizabeth J. Oakley, Publisher

PS Form 3526, Dec. 1985 *(See instructions on reverse)*

TRAPPED By Your ROM BIOS?

We can help you update your old PC, XT, AT and 386 machines. The latest ROM BIOS from Award Software supports:

* Enhanced Keyboards
* EGA & VGA Graphics
* Fast Processor Speeds
* 3.5" Micro Floppies... and more

Call 1-800-423-3400 KOMPUTERWERK, INC.
Your Authorized Award BIOS Distributor

FREE Catalog QuickBASIC Tools

Why shop around for hard-to-find BASIC tools and utilities? Our FREE catlog includes:

* The FINALLY! Family of Products
* Data Base & Graphics Libraries
* Window Management Libraries
* Screen Generators, Utilities... and more

Call 1-800-423-3400 KOMPUTERWERK, INC.
For Your FREE CATALOG

KOMPUTERWERK, INC
851 Parkview Blvd
Pittsburgh, PA 15215

Number 155 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash Code, Continued

6

```
        pop     es      ;ES:DX points to the color table
        int     10h     ;invoke the video interrupt to set the palette
;
; Put up the color numbers, so we can see how attributes map
; to color values, and so we can see how each color # looks
; (at least on this particular screen).
;
        call    ColorNumbersUp
;
; Wait for a keypress, so they can see this color set.
;
WaitKey:
        mov     ah,8    ;DOS input without echo function
        int     21h
;
; Advance to the next color set.
;
        mov     al,[CurrentColor]
        inc     ax
        mov     [CurrentColor],al
        cmp     al,64
        jbe     ColorLoop
;
; Restore text mode.
;
        mov     ax,3
        int     10h
;
; Done.
;
Done:
        mov     ah,4ch  ;DOS terminate function
        int     21h
;
; Puts up a bar consisting of the specified attribute (pixel value),
; at a vertical position corresponding to the attribute.
;
; Input: AL = attribute
;
BarUp   proc    near
        mov     dx,SC_INDEX
        mov     ah,al
        mov     al,MAP_MASK
        out     dx,al
        inc     dx
        mov     al,ah
        out     dx,al   ;set the Map Mask register to produce
                        ; the desired color
        mov     ah,BAR_HEIGHT
        mul     ah      ;row of top of bar
        add     ax,TOP_BAR ;start a few lines down to leave room for
                           ; text
        mov     dx,80   ;rows are 80 bytes long
        mul     dx      ;offset in bytes of start of scan line bar
                        ; starts on
        add     ax,20   ;offset in bytes of upper left corner of bar
        mov     di,ax
        mov     ax,VGA_SEGMENT
        mov     es,ax   ;ES:DI points to offset of upper left
                        ; corner of bar
        mov     dx,BAR_HEIGHT
        mov     al,0ffh
BarLineLoop:
        mov     cx,40   ;make the bars 40 wide
        rep     stosb   ;do one scan line of the bar
        add     di,40   ;point to the start of the next scan line
                        ; of the bar
        dec     dx
        jnz     BarLineLoop
        ret
BarUp   endp
;
; Converts AL to a hex digit in the range 0-F.
;
BinToHexDigit   proc    near
        cmp     al,9
        ja      IsHex
```

7

```
        add     al,'0'
        ret
IsHex:
        add     al,'A'-10
        ret
BinToHexDigit   endp
;
; Displays the color values generated by the color bars given the
; current palette register settings off to the right of the color
; bars.
;
ColorNumbersUp  proc    near
        mov     ah,2    ;video interrupt set cursor position function
        sub     bh,bh   ;page 0
        mov     dh,TOP_BAR/14   ;counting in character rows, match to
                                ; top of first bar, counting in
                                ; scan lines
        mov     dl,20+40-1      ;just to right of bars
        int     10h
        mov     al,[CurrentColor]       ;start with the current color
        mov     bx,offset ColorNumbers-1
                        ;build the color number text string on the fly
        mov     cx,16   ;we've got 16 colors to do
ColorNumberLoop:
        push    ax              ;save the color #
        and     al,3fh          ;limit to 6-bit color values
        shr     al,1
        shr     al,1
        shr     al,1
        shr     al,1            ;isolate the high nibble of the
                                ; color #
        call    BinToHexDigit   ;convert the high color # nibble
        mov     [bx],al         ; and put it into the text
        pop     ax              ;get back the color #
        push    ax              ;save the color #
        and     al,0fh          ;isolate the low color # nibble
        call    BinToHexDigit   ;convert the low nibble of the
                                ; color # to ASCII
        mov     [bx+1],al       ; and put it into the text
        add     bx,COLOR_ENTRY_LENGTH   ;point to the next entry
        pop     ax              ;get back the color #
        inc     ax              ;next color #
        loop    ColorNumberLoop
        mov     ah,9            ;DOS print string function
        mov     dx,offset ColorNumbers
        int     21h             ;put up the attribute numbers
        ret
ColorNumbersUp  endp
;
Start   endp
Code    ends
        end     Start


;
; *** Listing 4 ***
;
; Program to demonstrate screen blanking via bit 5 of the
; Attribute Controller Index register.
;
AC_INDEX        equ     3c0h    ;Attribute Controller Index register
INPUT_STATUS_1  equ     3dah    ;color-mode address of the Input
                                ; Status 1 register
;
; Macro to wait for and clear the next keypress.
;
WAIT_KEY        macro
        mov     ah,8    ;DOS input without echo function
        int     21h
        endm
;
stack   segment para stack 'STACK'
        db      512 dup (?)
stack   ends
;
Data    segment word 'DATA'
SampleText db   'This is bit-mapped text, drawn in hi-res '
```


PRODUCED 2042 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        db      'EGA graphics mode 10h.', 0dh, 0ah, 0ah
        db      'Press any key to blank the screen, then '
        db      'any key to unblank it.', 0dh, 0ah
        db      'then any key to end.$'
Data    ends
;
Code    segment                 cs:Code, ds:Data
        assume
Start   proc    near
        mov     ax,Data
        mov     ds,ax
;
; Go to hi-res graphics mode.
;
        mov     ax,10h  ;AH = 0 means mode set, AL = 10h selects
                        ; hi-res graphics mode
        int     10h     ;BIOS video interrupt
;
; Put up some text, so the screen isn't empty.
;
        mov     ah,9    ;DOS print string function
        mov     dx,offset SampleText
        int     21h
;
        WAIT_KEY
;
; Blank the screen.
;
        mov     dx,INPUT_STATUS_1
        in      al,dx   ;reset port 3c0h to index (rather than data)
                        ; mode
        mov     dx,AC_INDEX
        sub     al,al   ;make bit 5 zero...
        out     dx,al   ;...which blanks the screen
;
        WAIT_KEY
;
; Unblank the screen.
;
        mov     dx,INPUT_STATUS_1
        in      al,dx   ;reset port 3c0h to index (rather than data)
                        ; mode
        mov     dx,AC_INDEX
        mov     al,20h  ;make bit 5 one...
        out     dx,al   ;...which unblanks the screen
;
        WAIT_KEY
;
; Restore text mode.
;
        mov     ax,2
        int     10h
;
; Done.
;
Done:   mov     ah,4ch  ;DOS terminate function
        int     21h
Start   endp
Code    ends
        end     Start
;
; *** Listing 5 ***
;
; Solution to MASM 5.0 problem with local labels in REPT blocks.
; Brad Levy's solution (also offered by Michael Liebert).
;
        dosseg
        .model  small
        .code
maci    macro
        local   testlabel
        jmp     testlabel
testlabel:
        endm
```

Code continues . . .

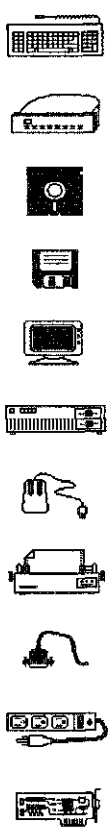
ICON-TOOLS...

from Ithaca Street Software, Inc.

C DEVELOPERS TAKE NOTE!

Now you can add icons to any C graphics application quickly and easily with ICON-TOOLS, an icon display and editing system for the IBM PC, PS/2, and all compatibles. Consider the features:

* Powerful, full-featured, mouse-driven icon editor. Cut and paste, construct shapes, copy and import. Many other features.
* Icon capture utility. Display and clip icons from popular graphics file formats, including .PCX, .IMG, and .MSP.
* Icon file modify utility. Resize icons to any width or height (not just multiples of the original size). Smooth icons with a convolution filter (get rid of the "jaggies").
* C source code for a complete icon function library. Separate versions included for 5 popular graphics packages: Turbo C BGI, MS Quick C, Essential Graphics, MetaWINDOWS, and C Source's GFX.
* 20 sample icon files containing over 500 different icons. Use as-is or modify for your own needs. (The icons at right are from our sample files.)
* No royalties, and a money-back guarantee.



ONLY $69.95 (After Jan 1, 1989 price will be $79.95). Take advantage of this special price and order today.

Ithaca Street Software, Inc.
1145 Ithaca Dr, Boulder, CO 80303 (303) 494-8865

Number **226** on the Reader Service Card

SPECIAL REPORT

COMPUTER VIRUS PROTECTION!

The threat of computer viruses is real!

FILE INTEGRITY CHECKing system performs a thorough check of all of the programs and files on your hard disk(s). Its complete reporting methods can alert you to the potential presence of a virus in your system. Works on all IBM PC, XT, AT, PS/2 and true compatibles running DOS.

Our checking methods include:

* Subdirectory names * Added/Deleted/Renamed Files/Dirs
* Filenames * Attributes * Time * Date * Size * CRC * MCRC
* Boot Records/Partition Table * Thorough, complete reports

Our unique MCRC (Modified CRC) check was developed because a virus could possibly alter a file yet keep the same CRC - if that happens, our MCRC check would catch it! Nothing is more effective than our dual checking methods.

The best part is the price: $15. Not much to pay for peace of mind. In addition, the FILE INTEGRITY CHECKing system comes with six months of free access to our BBS which deals with the topic of computer viruses 24 hour a day! Send your check today or order by phone with your MASTER CARD or VISA. (California residents please add 6 1/2% sales tax.)

GILMORE SYSTEMS
213-275-8006 Download a FREE evaluation copy from our BBS @ 213-276-5263
P.O. Box 3831, Beverly Hills, CA 90212-0831

Number **135** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash Code, Continued

```
mac2    macro
        rept    10
        mac1
        endm
        endm

        mac2

        end

; *** Listing 6 ***
;
; Solution to MASM 5.0 problem with local labels in REPT blocks.
; Adrian Crum's solution.
;
; Note: this code is relatively simple, it should be easy to follow.
; Make sure that the counter, (?CNT1) is initialized only one in
; your program. Otherwise, you will receive multi-defined errors.
;
DOSSEG
.MODEL SMALL
.CODE
;
SYM_SUPP1       MACRO   X
;; support macro #1 for SYMBOLS macro
;;
        jmp     ?TMPLBL&X&
;;
;; Note: if you're sure that these will always be short jumps,
;; then JMP SHORT ?TMPLBL&X&
;;
ENDM
SYM_SUPP2       MACRO   X
;; Support macro #2 for SYMBOLS macro
;;
?TMPLBL&X&:
;;
ENDM
;
SYMBOLS MACRO
;; This is the main macro. No calling parameters. This macro
;; calls two other macros to generate 10 JMP instructions to
;; 10 temporary labels and the temporary labels themselves.
;;
        REPT    10
                SYM_SUPP1       %?CNT1  ;; Generate JMP instruction
                SYM_SUPP2       %?CNT1  ;; Generate label
                ?CNT1 = ?CNT1 + 1       ;; Increment counter
        ENDM
;;
;; Note: you could also define the number of repeats executed
;; by this macro by including a dummy parameter X at the
;; beginning of the macro and changing REPT 10 to REPT &X. The
;; correct call in that case would be SYMBOLS n, where n is
;; the number of repetitions.
;;
ENDM
;
START:
?CNT1 = 0                       ; INITIALIZE COUNTER
SYMBOLS                         ; GENERATE 10 TEMP LABELS
SYMBOLS                         ; GENERATE 10 MORE LABELS
REPT 3                          ; GENERATE 3 MORE LABELS
        SYM_SUPP1       %?CNT1
        SYM_SUPP2       %?CNT1
        ?CNT1 = ?CNT1 + 1
ENDM
;
END     START
```


Tech Help!

The Ultimate Electronic Reference Manual

Your printed manuals are obsolete!

Tech *Help!* is a unique hypertext encyclopedia.
Pop up a Megabyte of technical information without leaving your editor or debugger.

- DOS 3.3 functions and all ROM BIOS services — every detail!
- Every known PC/XT/AT data structure, including FCB, PSP, boot sector, partition table, FAT, EXEC control block, and more!
- Keyboard scan codes, Extended ASCII keystrokes, ANSI.SYS escape codes, ASCII tables, BIOS data map, and MUCH more!
- 8088/8086/80286 Instruction Set
- LIM Expanded Memory Specification
- DOS, BIOS, and hardware information in one hypertext source!
- Same engine as in DOS Help! (PC Magazine "Editor's Choice")
- Uses 62K of RAM when resident; 1 Megabyte of compressed text occupies only 340K of your disk. Hard disk recommended.

Written by **Dan Rollins**,
author of the Norton Online Guide to OS/2.

Order now—check, money order, or MC/Visa
Only $89.95 + $4 S&H. Full money-back guarantee!
Toll free order line . . . ask for extension **1233**

800-833-7355
in California call **800-358-4477**

Flambeaux
1147 E. Broadway #56, Glendale, CA 91205 (818) 500-0044

Number **206** on the Reader Service Card

Celebrating our 10th Anniversary!

AMX™ 68000

KADAK's engineers bring years of practical real-time experience to over 600 installations world-wide.

This *real time* MULTITASKING KERNEL simplifies *real life* product development

- No royalties
- Full source code included
- * C interface
- Preemptive scheduler
- Intertask messages
- Terminal Handler
- * Dynamic memory allocation
- * Event Manager
- * Semaphore Manager
- * Resource Manager
- * Buffer Manager

AMX 68000 operates on any 68000/10/20 system

Manual only **$75 US**
AMX 68000 **$1600 US**
* Options — ask for price list.
(Shipping/handling extra)

Also available for 8080, Z80, 8086/186/286.

KADAK Products Ltd.
206 - 1847 W. Broadway
Vancouver, B.C. Canada V6J 1Y5
Telex: 04-55670
Fax: (604) 734-8114
Telephone: (604) 734-2796

Number **211** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

64K Extra Memory for DOS Applications

by
Jake Richter

When Microsoft announced Windows/286 this past spring, there was a little surprise attached. Microsoft had "found" almost 64K more of memory for Windows/286 to use on AT compatibles with at least 1MB of memory. In fact, Microsoft used a little known feature of the 80286 and 80386 chips.

When Intel designed the 80286 and 80386 processors, they had to make compromises that resulted in some incompatibility with the 8086.

One incompatibility was the lack of wrap-around when accessing the high end of the first megabyte of processor memory while in real mode. On the 8086 (and 8088), if you read the memory at address FFFF:0010 (in standard **segment:offset** notation), you were actually reading memory from 0000:0000 because FFFF:0010 translated to a linear address of 100000 Hex; and since the 8086 had only a 20 bit address space, just the low 5 nibbles were observed by the processor, and thereby the 21st address bit (address bit 20) was ignored (i.e., 100000 Hex became 00000 Hex).

This changed with the advent of the 80286, which had 24 bits of address space. No wrap occurred when reaching the top end of the first megabyte, even in real mode.

IBM apparently decided that this was not acceptable and implemented in hardware an override (gate) for Address Line 20 (known as A20). When this override was active, the A20 line coming out of the 80286 was ignored and the A20 line going to the memory bus was set low (zero). But since A20 was necessary in protected mode, IBM had to provide the means to disable the override and thus enable the A20 line from the processor. Just about all AT compatibles, including those based on the 80386, have this capability.

With A20 enabled, an 80286/80386 user with at least 64K of extended memory has access to another 65520 bytes of memory in real mode. The number of bytes is derived by subtracting 16 from 64K because 16 bytes are unusable since the largest segment value that can be used to access this additional memory is FFFF Hex.

Note that INT 15H in the BIOS modifies the state of the A20 Gate when it accesses extended memory. Also, using this additional memory will probably conflict with LIM, EMS, and ramdisks if they are installed.

I have provided a small piece of assembler code that will enable recognition of the A20 line. Use the well-known MASM-LINK-EXE2BIN sequence to produce a COM file. ♦

Jake Richter is president of Panacea Inc., a PC consulting firm. Readers may contact Jake on MCI Mail or BIX, user name JRICHTER.

Code follows.

```
;-----------------------------------------------------------------------------
; EnabA20.asm - by Jake Richter, Panacea Inc. 07/13/88
;
; This routine enables Address Line 20 from the 80X86 processor when making
; memory references. Default is to drive A20 low, i.e. A20 disabled.
; The line state is controlled by the 8042 Keyboard Controller.
;
; Made executable with the following sequence:
;
; MASM EnabA20;
; LINK EnabA20;
; EXE2BIN EnabA20 EnabA20.com
;
;
COM8042     equ     064H
DAT8042     equ     060H



Cseg        segment
            assume  CS:Cseg, DS:Cseg
            org     100H                    ; Needed for COM file.
Gate20      proc    near
            cli                             ; Disable interrupts.
            call    WaitOn8042             ; Make sure 8042 input buff is empty.
            jnz     Gate20_Ret              ; Leave if command not accepted.
            mov     AL, 0D1H                ; 8042 "Write Output Port" command.
            out     COM8042, AL             ; Send command to 8042.
            call    WaitOn8042             ; Wait for 8042 to ACK the command.
            jnz     Gate20_Ret              ; If it doesn't, exit.
            mov     AL, 0DFH                ; Grab A20 enable value.
                                            ; To disable, use 0DDH instead.
            out     DAT8042, AL             ; Send it.
            call    WaitOn8042             ; Wait for 8042 one last time.
            sti                             ; Restore interrupts.
Gate20_Ret:
;           cmp     AL, 0                   ; Implement if error check
;                                           ; is needed...
;           jne     DidNotWork
            ret                             ; Return to DOS.
Gate20      endp

WaitOn8042  proc    near
            xor     CX, CX                  ; Clear timeout counter.
WaitLoop:
            in      AL, COM8042             ; Read 8042 status.
            and     AL, 02                  ; Test for input buffer full.
            loopnz  WaitLoop                ; Wait until timeout or OK.
            ret
WaitOn8042  endp

Cseg        ends
            end ♦
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOS Device Drivers

A Debugging Utility and Definition

by
Stephen A. Rodgers

History

Version 1.0 of DOS had no provision for installable device drivers. All device-dependent code was contained within a module in DOS. These "pre-installed" device drivers made it a formidable task to install add-on devices such as external hard disks. Users were stuck with the device complement supplied by the manufacturer of the computer.

Installable device driver support started with DOS 2.0. Installable device drivers were to allow users and third party hardware vendors to write special purpose device drivers for hardware not supported by DOS, or to replace a pre-installed device driver with an enhanced version (such as an interrupt-driven COM driver).

Later releases of DOS enhanced the device driver concept by adding more functions to the device driver interface. DOS 3.0 added open/close/removable media bit, and DOS 3.2 added the generic I/O request that made it possible to format hard disks through the device driver interface. Tables 1 and 2 show the history of device level function codes and IOCTL subfunction calls.

Debugging a Device Driver

One of the most difficult tasks a developer can undertake is debugging a device driver. Unless you want to add a large amount of debug code to a driver, you don't know where DOS is going to load the driver into memory. A utility that would display this information would be very useful, allowing you to debug a device driver within the DOS environment.

Initially, a device driver should be debugged outside the DOS environment before it's installed in CONFIG.SYS. This is best accomplished by writing a front end for the device driver that builds request headers and issues far calls to the strategy and interrupt procedures. This is especially true when debugging the INIT procedure within the device driver since a successful load depends on the proper operation of this code.

How DOS Installs Device Drivers

DOS loads installable device drivers when it is performing boot-up or SYSINIT. The order in which DOS installs a particular device driver is determined by the block/character attribute in the device header. If the attribute is set to block, DOS installs the device in the chain following the base device drivers in the BIOS module. If the at-

Table 1. Device Level Function Codes

		DOS Version Numbers					
Function		2.0	2.1	3.0	3.1	3.2	3.3
0	INIT	*	*	*	*	*	*
1	MEDIA CHECK	*	*	*	*	*	*
2	BUILD BPB	*	*	*	*	*	*
3	IOCTL INPUT	*	*	*	*	*	*
4	INPUT	*	*	*	*	*	*
5	NONDEST. INPUT	*	*	*	*	*	*
6	INPUT STATUS	*	*	*	*	*	*
7	INPUT FLUSH	*	*	*	*	*	*
8	OUTPUT	*	*	*	*	*	*
9	OUTPUT VERIFY	*	*	*	*	*	*
10	OUTPUT STATUS	*	*	*	*	*	*
11	OUTPUT FLUSH	*	*	*	*	*	*
12	IOCTL OUTPUT	*	*	*	*	*	*
13	DEVICE OPEN			*	*	*	*
14	DEVICE CLOSE			*	*	*	*
15	REMOVABLE MEDIA			*	*	*	*
16	OUTPUT UNTIL BUSY			*	*	*	*
19	GENERIC IOCTL					*	*
23	GET LOGICAL DEV.					*	*

Table 2. IOCTL Subfunction Codes

	DOS Version Numbers					
AL	2.0	2.1	3.0	3.1	3.2	3.3
00H GET DEVICE INFORMATION	*	*	*	*	*	*
01H SET DEVICE INFORMATION	*	*	*	*	*	*
02H CHARACTER DEVICE READ	*	*	*	*	*	*
03H CHARACTER DEVICE WRITE	*	*	*	*	*	*
04H BLOCK DEVICE READ	*	*	*	*	*	*
05H BLOCK DEVICE WRITE	*	*	*	*	*	*
06H GET INPUT STATUS	*	*	*	*	*	*
07H GET OUTPUT STATUS	*	*	*	*	*	*
08H REMOVABLE MEDIA TEST			*	*	*	*
09H REMOTE DEVICE TEST				*	*	*
0AH REMOTE HANDLE TEST				*	*	*
0BH ADJUST SHARING RETRIES				*	*	*
0CH CHARACTER GENERIC IOCTL					*	*
0DH BLOCK GENERIC IOCTL					*	*
0DH GET LOGICAL DEVICE MAP					*	*
0DH SET LOGICAL DEVICE MAP					*	*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Table 3. DDI Device Attribute Bits

BIT	DESCRIPTION	DOS Version Numbers 2.0	2.1	3.0	3.1	3.2	3.3
15	CHAR	*	*	*	*	*	*
14	IOCTL	*	*	*	*	*	*
13	NONIBM/OUTUNTILBSY	*	*	*	*	*	*
11	OPNCLS			*	*	*	*
6	LOGDEV/GENERICIOCTL					*	*
3	STDCLK	*	*	*	*	*	*
2	NULDEV	*	*	*	*	*	*
1	STDOUT	*	*	*	*	*	*
0	STDIN	*	*	*	*	*	*

tribute is set to character, DOS installs the device ahead of all the pre-installed devices so that it may replace a pre-installed device if it has the same name. If character devices are loaded ahead of all other devices in the chain, the last character device specified in CONFIG.SYS will be one away from the head of the device list. The NUL device driver, contained within DOS, is at the head of the device list and therefore can't be replaced.

An installable device driver contains a device header at the beginning of the executable image. This device header contains several pointers, one of which is a pointer to the next device in the chain. Initially, this pointer is equal to –1, or the offset portion points to the next driver in the image. During SYSINIT, DOS adjusts each pointer to point to the next device in the chain. If it is at the last device, DOS leaves the next device pointer initialized to –1 (FFFF:FFFF).

An Information Utility—DDI

Using this information, a linear search can be initiated starting at address 0060:0000. A device name is always 8 characters in length and padded with trailing blanks. If we check for a match with the first character (in this case with an "N") and then check for subsequent character matches, the NUL device will be found somewhere within segment 0060. The position of the NUL device in the DOS image is dependent upon the version of DOS and upon the OEM who customized it.

The information contained in the device header of the NUL device provides a means to "trace" through the device chain since it is essentially at the head of a linked list. If we follow the pointers in all of the device headers, we will be able to trace through the entire device chain, allowing us to locate any block or character device. The end of the device chain is denoted by a pointer having the value FFFF:FFFF.

For block devices, a different approach is taken. Undocumented DOS function call 32H will return the address of the device driver responsible for a particular drive. This function is used to obtain the address of the desired block device.

Using DDI

As mentioned above, DDI has two modes of operation. The first mode displays all loaded device drivers as they are actually chained in the system. The second mode displays detailed information about a specific device. The second mode decodes all attribute bits (see Table 3) and displays the strategy and interrupt entry points. For example, to get information on the CON device, type

```
DDI CON
```

Block devices may be displayed by using the appropriate drive letter. For example, to get the device driver information for the C: drive, type

```
DDI C:
```

♦

Stephen A. Rodgers is Chief Engineer for Innovative Data Technology, a manufacturer of 9 track tape drives and 9 track PC tape subsystems.

Code follows.

Did You Know . . .

The size of the punched card was based on the size of the old-style U.S. paper currency (it was supposed to be a good size for convenient handling.)?—*W.G. Madison* ♦

Don't let C bugs waste YOUR time

Uncover C coding errors automatically with PC-lint

Make your C programs more portable, more maintainable and more manageable. Use PC-lint to analyze your C programs (one or many modules) and uncover glitches, bugs, quirks, and inconsistencies that your compiler, working on one module at a time, will miss. *"PC-lint has everything going for it: flexibility, speed, good documentation, and a reasonable price. -- Stephen D. Cooper, San Francisco PC Users Group*

PC-lint supports full K&R C plus ANSI extensions. *"... a remarkable well thought-out product which will check for just about every conceivable coding error ... we confidently recommend PC-lint." -- Andrew Binstock, The C Gazette*

Customize PC-lint to your own programming style and project. Turn off any error message locally or globally, alter size of scalars, adjust format of error messages, and more. *"... friendliness is a prime consideration in PC-lint. Gimpel has provided ways to make Lint shut up about all those errors you either know or don't care about. -- Don Malpass, IEEE Software*

NOW AVAILABLE -- Generic Lint in shrouded source form for use on VAX/VMS, Unix, OS-9, IBM VM/CMS - MVS, etc. Requires only K&R C to compile. Pricing starts at $798. Call for details.

(215)584-4261

Only $139 - CALL TODAY

30 Day Money-back Guarantee

Gimpel Software

3207 Hogarth Lane

Collegeville PA 19426

Specify MS-DOS or OS/2. Quantity and educational discounts available.

PA add 6% tax -- Outside US add $20.

PC-lint and Generic Lint are trademarks of Gimpel Software.

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk.
ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386.
Mail order information on pages 8 and 88.

Code from "DOS Device Drivers" by Stephen A. Rodgers

1

```
/* DDI - Device information program
*
* Written By:   Steve Rodgers
* Conception:   5-4-86
*
* Copyright (C) 1987 Steve Rodgers
*
* Functional Description
* ----------------------
*
* This program displays information about installable device
* drivers loaded under DOS 2.0 or later. If the command is typed
* without any arguments, a summary will be printed that shows
* the order of the devices in the chain. If a device name is
* typed following the command, detailed information pertinent to
* that device will be displayed. This program works with either
* character or block devices as arguments.
*
* Use the Microsoft C compiler Version 3.00 or later.
* Compile with the -Zp switch.
*
* Use the Microsoft linker Version 3.00 or later.
*/
#include <stdio.h>
#include <dos.h>
#include <string.h>

/* Device attribute definitions */

#define CHARACTER_DEVICE        0x8000
#define IOCTL_SUPPORT           0x4000
#define NON_IBM_FORMAT          0x2000
#define OPEN_CLOSE_SUPPORT      0x800
#define IS_CLOCK                8
#define IS_NUL                  4
#define IS_STDIN                2
#define IS_STDOUT               1

/* Structure definitions--Device header */
struct DEV_HDR
{
    struct DEV_HDR far      *next_hdr ;     /* pointer next device header */
    unsigned int            attribute ;     /* attribute bits */
    unsigned int            strategy ;      /* strategy offset */
    unsigned int            interrupt ;     /* interrupt offset */
    unsigned char           name[8] ;       /* device name */
} ;

/* Disk Parameter Block - Returned by DOS function 32H */

struct DPB
{
    unsigned char           ad ;            /* current assigned disk */
    unsigned char           alt_ad ;        /* alternate assigned disk */
    unsigned int            bps ;           /* bytes per sector */
    unsigned char           last_sic ;      /* max sector addr in a cluster */
    unsigned char           last_head ;     /* max useable head address */
    unsigned int            rs ;            /* total reserved sectors */
    unsigned char           cf ;            /* copies of the FAT */
    unsigned int            d ;             /* max root directory entries */
    unsigned int            fus ;           /* first useable sector */
    unsigned int            tcoplusone ;    /* total clusters available + 1 */
    unsigned char           spf ;           /* sectors per FAT */
    unsigned int            fds ;           /* first directory sector */
    struct DEV_HDR far      *dda ;          /* device driver address */
    unsigned char           md ;            /* media discriptor */
    struct DISK_DPB far     *next_dpb ;     /* address of next dpb in chain */
    unsigned int            cwd_cluster ;   /* cluster of CWD (DOS 2.x only) */
    unsigned char           cwd_string[64] ; /* CWD string (DOS 2.x only) */
```

2

```
/* Function declarations */

struct DPB far *disk_dpb() ;
struct DEV_HDR far *device_chain() ;

/* Static Declarations */

union REGS ir,or ;
struct SREGS sr ;

int is_disk = 0 ;

/* Attribute bit table */

int attr_bit_masks[] = {CHARACTER_DEVICE,IOCTL_SUPPORT,NON_IBM_FORMAT,
                        OPEN_CLOSE_SUPPORT,IS_CLOCK,IS_NUL,IS_STDIN,IS_STDOUT};

/* Global pointer variables */

struct DEV_HDR far *current_device, far *previous_device ;
struct DPB far *current_dpb ;

/* Global device type counters */

int block_drivers = 0,character_drivers = 0 ;

/* Function: main() */

main(argc,argv)
int argc ;
char *argv[] ;
{
    int i ;
    char *device_name ;

    printf("\nDDI    -              Device Driver Information Program\n") ;
    printf("Version 1.0 (2-18-88) Copyright (C) 1988 Steve Rodgers\n\n") ;

    if(argc < 2) /* if no argument, print the device list */
    {
        list_devices() ;
        exit(0) ;
    }

    if((device_name = strupr(strdup(argv[1]))) == NULL)
        abort() ;
    print_single_device(device_name) ;
}

/* Print information for a single device. */

print_single_device(device_name)
char device_name[] ;
{
    int i ;

/* Get pointer for a specific device.
* If it doesn't exist print an error message. */

    get_device(device_name) ;
    if(current_device == NULL)
    {
        printf("No such device: %s\n",device_name) ;
        exit(1) ;
    }

/* Print the device address, type, and name. */

    print_device_columns() ;
    print_device_info(current_device) ;

/* Print the detailed information for the device. */
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Journal
The Resource Journal for IBM Micro Programmers

P. O. Box 30160 • Eugene, OR 97403 • (503) 747-0800

January 5, 1989

Dear *PJ* Reader:

As you've probably noticed by now, page 38 of the January/February 1989 *Programmer's Journal* is missing a column of text. Mistakes like this are easily and often blamed on "printer error." Well, this time it *wasn't* our fault: Our printer really did goof, and they are working with us to make it up to you.

Although SLR Systems is no doubt thrilled to have their ad appear twice in one issue, we're sure you would rather have the material from Jake Richter's article. Therefore, we are having the printer send you a copy of the corrected page on the back of this letter. (If it's not there, they are *really* in trouble.)

We apologize for any inconvenience this blunder may have caused you. If we're all lucky, however, you may have a collector's item on your hands: The issue may contain the last mistake you ever find in *PJ*. (I know . . . and pigs may learn to fly in 1989, too.)

Sincerely yours,

Robert E. Wanetick

Robert E. Wanetick
Managing Editor

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A GP command sequence is like a program, but in a very simple language. You load a string of GP commands into the 82786 memory, terminate it with an NOP that has its ECL (End of Command Line) bit set, and then tell the GP to start executing from a specific point in memory. Since the chip does not have any conditional branch instructions, its programmability is severely limited. It does have CALL and RETURN commands, though; so you could set up a generic command list to, for example, draw a box with an *X* in it at the current *x,y* position and make it a subroutine that you could call in from your 82786 "program."

Table 1. Logical Writing/Pixel Processing Mode on the 34010 (The first 16 are also supported by the 82786.)

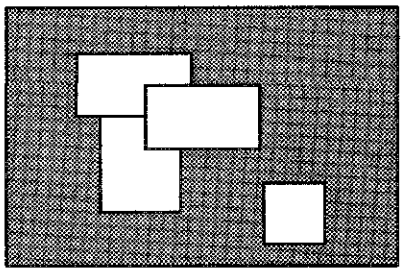
Definitions:

0s	All bits in the pixel are set to 0
1s	All bits in the pixel are set to 1
DST	The pixel at the destination
SRC	The source pixel
AND	Bitwise AND operation
OR	Bitwise OR operation
XOR	Bitwise XOR operation
NOT	Bitwise NOT (one's complement) operation
ADD	Unsigned Addition
SUB	Unsigned Subtraction
ADDS	ADD with Saturation (no wrap, max's out at max pixel value)
SUBS	SUB with Saturation (no wrap, stops at a pixel value of 0)
MAX	Larger of the two operands
MIN	Smaller of the two operands

The result of a pixel processing operation is placed in DST.

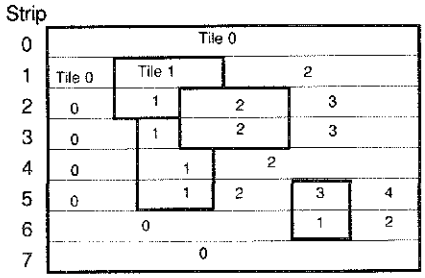
Processing Modes:

```
SRC
SRC AND DST
SRC AND (NOT DST)
0s
SRC OR (NOT DST)
(NOT SRC) XOR DST
NOT DST
(NOT SRC) OR (NOT DST)
SRC OR DST
DST
SRC XOR DST
(NOT SRC) AND DST
1s
(NOT SRC) OR DST
(NOT SRC) AND (NOT DST)
NOT SRC
SRC ADD DST
SRC ADDS DST
DST SUB SRC
DST SUBS SRC
MAX(SRC, DST)
MIN(SRC, DST)
```



Shaded area is field, i.e., not data.

What you see on screen, 4 hardware windows



8 strips, with 1 to 5 tiles each

Strip and tile breakdown of an 82786 screen

The Display Processor is probably the most underutilized feature of the chip. Most graphics-board companies using the 82786 use the DP for driving the monitor output and not much else. But the DP provides two other features: a hardware cursor and hardware windowing.

Hardware windows are similar in concept to the overlapping software windows utilized in such packages as Microsoft Windows. The big difference is that hardware windows do not require a redraw of screen information when the window is moved. This means that you can have virtually simultaneous drawing going on in a number of overlapping windows without worrying about clipping, because the hardware windows only display what's in memory; they do not alter memory. Hardware windows can also display *field*, a term used to indicate non–memory consuming display information ("dead space"). Because all normal limitations are removed, the long-time tie between memory and display is completely altered, requiring programmers to toss out everything they know about displaying PC-based graphics.

Of course, you can still use the one-to-one correlation between memory and display by creating a full-screen hardware window. Hardware windows are rectangular tiles that display data from memory at 1, 2, 4, or 8 bits per pixel. To top that off, a window can be flagged to indicate to the hardware that it should be zoomed. The zoom factor ranges from 1 to 64 in both the *x* and *y* directions. The screen position of a hardware window as well as the contents are determined by the DP display list. This display list consists of a series of strips, each of which contains a number of tiles. There is a limit of 15 tiles per strip, but no limit on the number of strips, providing each one is at least one scan line high.

Although the 82786 is a coprocessor, it does have three sets of registers that can be accessed by an external processor. One set belongs to the DP and is used for setting display frequencies, display list location, cursor pattern, and so forth. Another set belongs to the GP and contains information regarding the current drawing environment, such as *x,y* position, colors, and start of drawing area in memory. The last set of registers is the set containing the processor control. This set can be either memory

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Microsoft C Users

Lattice C Users

Frustrated by Long Development Cycles? Constrained by 640K Limitations?

"It's the best of both worlds for the experienced C programmer."
- PC Magazine, September 13, 1988

"...the best product which has ever been written for next generation PC's."
- David Kuechler, Vice President R&D, Program Technologies

Instant-C/16M™ is your solution.

"Unique and highly responsive C programming environment, ... ideally suited for developing C application prototypes."
- PC Week, August 1, 1988

"...best program development environment available anywhere."
- Karim Lakhani, Vice President Technical Development, Fast Trax

Instant-C/16M is the perfect complement to the Microsoft C and Lattice C compilers. You can develop your Microsoft C or Lattice C program in *Instant-C/16M* and save more than half of your development time. Once your program has been completely debugged with *Instant-C/16M*, compile it with Microsoft C or Lattice C for smallest code size and optimal code performance. You get the best of both worlds: fast development and superb debugging from *Instant-C/16M* and fast code from your current C compiler. Development benchmarks consistently show that *Instant-C/16M* is at least **twice as fast** as either Turbo C 1.5, Microsoft C 5.1, or QuickC 1.0 when comparing the turnaround times for compiling and linking.

Instant-C/16M was designed to give the professional programmer the ability to **make much larger programs -- much faster.** By providing access to the full 16MB of extended memory on 80286/386-based PCs, *Instant-C/16M* lets you create any size program you wish with all the features you want. ***Instant-C/16M* is simply the best and fastest way to develop and debug your C programs.**

Instant-C/16M

Instant-C/16M is constructed around an incremental C compiler and includes: a source-level debugger, editor, run-time checker, and library source. ***Instant-C/16M*** lets you add all the features you want to your program without worrying about the 640K limit. Programs of 75,000 lines of code and more than 1MB of dynamically allocated memory are easily handled.

Instant-C/16M's debugging capabilities are unsurpassed. Programs too large for Codeview™ fit easily within ***Instant-C/16M*** to make debugging large programs much easier. Combine ***Instant-C/16M***'s automatic run-time checking with protected mode hardware's checking and you have a "safety net" for catching those particularly difficult bugs that might normally take you hours, days, or even weeks to find and correct.

Features and Benefits

* ***Large Program Development*** -- *Instant-C/16M* gives you access to 16MB of extended memory on 80286/386-based machines. Development of multi-megabyte programs under DOS is no longer a problem. -- **Develop and debug very large programs.**

* ***Instant compilation*** -- Whenever you make a change *Instant-C/16M*'s incremental compiler recompiles only the changed portions of your code, not your entire source file. A typical change takes only one half (.5) to 2 seconds. -- **Saves your valuable development time.**

* ***Instant linking*** -- *Instant-C/16M*'s incremental compiler automatically links your code. -- **Zero link time.**

* ***Integrated environment*** -- The compiler, debugger, editor, run-time checker, and source formatter are tied together. You no longer need to quit one tool to start up another one. And all of *Instant-C/16M*'s tools are just a single keystroke away. -- **Everything is in one package.**

* ***Superior debugging*** -- Source-level debugger automatically traps errors in: invalid pointer references, computed function addresses, overstored return addresses, out of bounds array references, invalid pointer differences, conversion errors, invalid shifts, and arithmetic exceptions. The debugger supports the full C language including local symbols and #defines. -- **The best debugger available for C today.**

* ***Supports your existing programs*** -- *Instant-C/16M* supports the Microsoft and Lattice C compilers and all of your current C source code and libraries. -- **Protects your software investment.**

Instant-C

Instant-C has all the same productivity benefits of *Instant-C/16M* and supports 8086 and 8088-based PCs and compatibles.

Instant-C is designed to run your program at speeds comparable to C compilers. In addition, it lets you generate stand-alone programs -- if you don't need the benefits of a traditional optimizing compiler, don't buy one. ***Instant-C*** can be the only development tool you use.

Just like *Instant-C/16M*, ***Instant-C*** offers you full support for Microsoft C 5.1 and Lattice C 3.2 and support for graphics boards and non-PC hardware.

Why waste any more of your development time? ***Instant-C*** lets you develop your code twice as fast.

The Optimal C Development System

Develop and Debug
in
Instant-C/16M
then
Deliver
with
Your Favorite C Compiler

For More Information Call or Write

Rational Systems, Inc.

P.O. Box 480
220 North Main Street
Natick, MA 01760
Tel: **(508) 653-6006**
FAX: **(508) 655-2753**

Unconditional 30 Day Money Back Guarantee!

Trademarks are acknowledged to Rational Systems, Inc., Microsoft Corp., Lattice, and Borland.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Rodgers Code, Continued

3

```
    printf("\n") ;
    printf("                A T T R I B U T E S\n") ;
    printf("-------------------------------------------------------\n") ;
    printf("| CHAR   | IOCTL  | NONIBM | OPNCLS | STDCLK | NULDEV | STDOUT | STDIN |\n") ;
    for(i = 0 ; i < 8 ; i++)
    {
         printf("   ") ;
         if((attr_bit_masks[i] & current_device->attribute))
              printf("yes   ") ;
         else
              printf("no    ") ;
    }
    printf("\n\n") ;
    printf("Strategy entry point\t\t: %04X:%04X\n", FP_SEG(current_device),
        current_device->strategy) ;

    printf("Interrupt entry point\t\t: %04X:%04X\n", FP_SEG(current_device),
        current_device->interrupt) ;

    exit(0) ; /* Return to DOS. */
}

/* Print a summary of all devices in the chain. */

list_devices()
{

/* Get the head of the device list.
* If this isn't possible, print an error message and exit. */

    if((current_device = device_chain()) == NULL)
        no_device_chain() ;

/* Print the column information. */

    print_device_columns() ;

/* Follow the chain until a -1 is reached. */

    while(FP_OFF(current_device) != -1)
    {
         print_device_info(current_device) ;
         current_device = current_device->next_hdr ;
    }
    printf("\nSystem has %d character drivers, and %d block drivers\n",
            character_drivers,block_drivers) ;
}

/* Find character or block device driver.  Fill in current device
* pointer, previous device pointer, and current DPB if block device. */

get_device(d)
char d[] ;
{
    int i ;
    char device_string[9] ;

    current_device = NULL ;
    current_dpb = NULL ;
    if(strlen(d) > 8)
         return ;

/* If there is a colon in the second character position, user
* is requesting information about a block device. */

    if(d[1] != ':')
    {

/* The name specified is probably a character device.
* If a colon follows the device name, delete it from the string. */

       if(d[strlen(d) - 1] == ':') /* kill colon at end of device name */
            d[strlen(d) - 1] = 0 ;

/* Attempt to get the pointer to the first device in the chain.  If it
* comes back NULL, the user probably did not install the trace device driver. */
```

4

```
       if((current_device = device_chain()) == NULL)
            no_device_chain() ;  /* no device chain available */

/* Trace the device chain until device name matches a device in the chain. */

       while(FP_OFF(current_device) != -1)
       {
           for(i = 0 ; i < 8 ; i++)
               device_string[i] = current_device->name[i] ;
               device_string[8] = 0 ;
               if(!strncmp(device_string,d,strlen(d)))
                   if((strlen(d) == 8) || (current_device->name[strlen(d)] == ' '))
                   return ;
               previous_device = current_device ;
               current_device = current_device->next_hdr ;
       }

/* If no match, set the current device to NULL and return. */

            current_device = NULL ;
            return ;
    }

/* A colon was found in the second character position of the device
* name. Check for a valid block device. */

    else
        is_disk = 1 ; /* set a flag to signify a disk device */

    if(strlen(d) > 2)  /* length must be less than or equal to 2 */
        return ;

/* Get the disk DPB for the drive specified. If the disk doesn't exist, the * current_dpb
pointer will be set to NULL. Print a message and exit if this is the case. */

    if((current_dpb = disk_dpb[d[0] - 0x40]) == NULL)
        no_disk_dpb() ;
    current_device = current_dpb->dda ;
    previous_device = NULL ;
    return ;
}

/* Print the headings and columns when listing all device drivers. */

print_device_columns()
{
    printf("Base Address\t\t\tType\t\t\tName/Units\n") ;
    printf("------------\t\t\t----\t\t\t----------\n") ;
}

/* This function prints detailed information for a specific device. */

print_device_info(device)
struct DEV_HDR far *device ;
{
    char device_string[9] ;
    int i ;
    printf("%04X:%04X\t\t\t",FP_SEG(device),FP_OFF(device)) ; /* print base address */
    if(device->attribute & CHARACTER_DEVICE)
    {
        ++character_drivers ;
        for(i = 0 ; i < 8 ; i++)
            device_string[i] = device->name[i] ; /* copy it into local data segment */
        device_string[8] = 0 ;
        printf("Char\t\t\t%s\n",device_string) ; /* print "Char" and name */
    }
    else
    {
        ++block_drivers ;
        printf("Block\t\t\t%u units\n",(int) device->name[0]) ;/* block dev.*/
    }
}
```

Code Continues . . .

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

5

```
/* Return the address of the Disk Parameter Block (DPB) for the drive
 * specified. This DOS function is undocumented in current DOS documentation. */

struct DPB far *disk_dpb(drive)
int drive ;
{
    char far *dsk_dpb ;

    ir.x.ax = 0x3200 ; /* Function call 32H */
    ir.h.dl = drive ;
    intdosx(&ir,&or,&sr) ;
    if(++or.h.al == 0)
        return (char far *) NULL ;
    FP_SEG(dsk_dpb) = sr.ds ;
    FP_OFF(dsk_dpb) = or.x.bx ;
    return (struct DPB far *) dsk_dpb ;
}

/* Return the address of the first device in the device chain. */

struct DEV_HDR far *device_chain()
{
    struct DEV_HDR far *dev_chain ;
    auto char far *search_ptr = (char far *) 0x00600000 ; /* start search at 0060:0000 */
    static char *nuldev ="NUL     " ;
    unsigned int x,i ;

    for(x = 0, i = 0 ; x < 65535 ; x++)
    {
        if(*search_ptr == *nuldev)    /* if first character matches, check others*/
        {
            for(i = 0 ; i < 8 ; i++)
            {
                if(*(search_ptr + i) != nuldev[i])
                    break ;
            }
            if(i == 8)      /* if i = 8 then the NUL device has been found */
                break ;
        }
        search_ptr++ ;
    }
    if(x == 0)
        return NULL ;

    search_ptr -= 10 ; /* get to device header  */
    return (struct DEV_HDR far *) search_ptr ;
}

/* Print message "Can't find device chain" and exit. */

no_device_chain()
{
    fprintf(stderr,"Can't find device chain\n") ;
    exit(1) ;
}

/* Print  message "Can't get DPB - Possible invalid drive\n" and exit. */

no_disk_dpb()
{
    fprintf(stderr,"Can't get DPB - Possible invalid drive\n") ;
    exit(1) ;
}
```


NEW VERSION 2

MICROSOFT, TURBO AND MIX POWER C PROGRAMMERS... C WINDOWS TOOLKIT™ PUTS YOU IN CHARGE OF VIDEO *and* DATA-ENTRY

NEW FEATURES

- *Comprehensive data-entry and verification procedures with scanf( ) type syntax.*
- *Predefined data-entry functions for dates, SSNs, telephone numbers, currency amounts with commas, time, scientific notation, etc.*
- *Turn any window into an editor with your choice of keyboard mapping (Wordstar™ by default).*
- *Virtual windows up to available memory.*
- *Save windows to disk.*
- *Nested windows.*
- *Create and edit VGA/MCGA/EGA/Hercules fonts of any size.*

WINDOWING FUNCTIONS
- Create pop-up windows
- Create pull-down menus
- Create spreadsheet menus
- Create context-sensitive help screens
- Store windows in RAM or on disk
- Modify windows out of view
- Use 8 different types of exploding windows
- Scroll and move windows

MicroWay 386 SUPPORT*
- Runs in protected mode
- Uses 32-bit instruction set

*Extra cost option-call

SYSTEM CONTROL
- Device-independent output
- Supports VGA/MCGA/EGA/Hercules/ CGA/MDA
- Detect the types of video adapter installed
- Switch between adapters
- Detect ANSI.SYS
- Control the size and position of the cursor
- Detect the enhanced keyboard
- Disable the video signal
- Delay program execution to microsecond resolution

VGA/EGA SUPPORT
- Use all 64 VGA/EGA colors
- Use VGA 29/43/50 line modes
- Use EGA 43-line mode
- Use 2 fonts simultaneously
- Design and edit custom fonts with FONTEDIT;* our font editor (included)
- Smooth scroll the screen
- Smooth pan the screen

FAST SCREEN I/O
- Write formatted output (like **printf( )**)
- Get snow-free output on the CGA
- Scroll the screen
- Use direct screen writes or the BIOS

HERCULES SUPPORT
- Create and edit your own Ramfonts™
- Detect the presence of a Hercules Card
- Detect Ramfont™ support
- Load and store Ramfonts™ to RAM or disk
- Switch between modes
- Detect whether the Hercules Card is in text or graphics mode

Full source code included — No run-time royalties.
Includes 230 page manual — 30-day Money-Back Guarantee

Magna Carta SOFTWARE

PO. Box 475594
Garland, TX 75047-5594
FAX: (214) 226-0386

Requires: IBM PC, XT,AT, PS/2 or compatible.
Supports Microsoft C 5.0/5.1/Quick C, Borland Turbo C 1.5, Mix Power C

CALL (214) 226-6909

Only $99.95
(Texas Residents add 8% sales tax.)

Number **212** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers)	$500
	Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
	CQL Query System (SQL retrievals plus windows)	$325
	GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$250
NEW!	Greenleaf Business Mathlib (exact decimal math, formatting, depreciation, interest, cash flow, statistics)	$250
	Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$220
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$175
	TurboTEX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTEX)	$170
NEW!	Sherlock (C debugging aid)	$170
	Essential resident C (TSRify C programs, DOS shared libraries)	$165
	Greenleaf Functions (296 useful C functions, all DOS services)	$160
	Essential C Utility Library (400 useful C functions)	$160
	Essential Communications Library (C functions for RS-232-based communication systems)	$160
	WKS Library Version 2.0 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	ME Version 2.0 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$130
	TurboGeometry (library of routines for computational geometry)	$125
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$115
	Minix Operating System (U**x-like operating system, includes manual)	$105
	PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
NEW!	Tele Operating System (TeleKernel, TeleWindows, TeleFile, & TeleBTree by Ken Berry)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
	Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	$80
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
NEW!	Polyglot Lisp-to-C Translator (includes Lisp interpreter, Prolog, and simple calculus prover)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$75
NEW!	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
	lp (flexible printer driver; most popular printers supported)	$65
	Quincy (interactive C interpreter)	$60
	PTree (parse tree management)	$60
NEW!	Icon-Tools (full-featured icon display and editing system)	$60
NEW!	Polyglot TSR Package (includes reminder, bookmark, virus catcher, cache manager, & speech generator)	$50
	MicroFirm Toolkit (28 Unixesque utilities for MS-DOS)	$50
	HELP! (pop-up help system builder)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
	Make (macros, all languages, built-in rules)	$50
	Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
	Coder's Prolog (inference engine for use with C programs)	$45
	Virtual Memory System (least recently used swapping)	$40
	C-Notes (pop-up help for C programmers ... add your own notes)	$40
NEW!	Heap I/O (treat all or part of a disk file as heap storage)	$40
	Biggerstaff's System Tools (multi-tasking window manager kit)	$40
	PC-XINU (Comer's XINU operating system for PC)	$35
	CLIPS (rule-based expert system generator, Version 4.2)	$35
	Tiny Curses (Berkeley curses package)	$35
NEW!	Polyglot RAM Disk (change disk size on the fly; includes utilities)	$30
	SP (spelling checker with dictionary and maintenance tools)	$30
	Clisp (Lisp interpreter with extensive internals documentation)	$30
	Translate Rules to C (YACC-like function generator for rule-based systems)	$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
NEW!	Pascal Compiler & Interpreter (P-codes, standard Pascal)	$25
	ICON (string and list processing language, Version 7)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX)	$25
	LEX (lexical analyzer generator; an oldie but a goodie)	$25
	Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
	AutoTrace (program tracer and memory trasher catcher)	$25
NEW!	Data Handling Utilities in C (data entry, validation & display; specify Turbo C or Microsoft)	$25
	Arrays for C (macro package to ease handling of arrays)	$25
NEW!	ANSI Forms (forms manager based on ANSI codes)	$20
	C Compiler Torture Test (checks a C compiler against K & R)	$20
	Benchmark Package (C compiler, PC hardware, and Unix system)	$20
	TN3270 (remote login to IBM VM/CMS as a 3270 terminal on a 3274 controller)	$20
	A68 (68000 cross-assembler)	$20
	List-Pac (C functions for lists, stacks, and queues)	$20
	XLT Macro Processor (general purpose text translator)	$20
NEW!	KeySwap (swaps CapsLock and CRTL key functions on XTs and ATs)	$20
NEW!	OOPS (collection of handy C++ classes by Keith Gorlen of NIH)	$20
	C/reativity (Eliza-based notetaker)	$15
	Data	
	DNA Sequences (GenBank 52.0 including fast similarity search program)	$150
	Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$60
	Dictionary Words (234,932 words in alphabetical order, no definitions)	$60
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
	KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
	USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	$20
	NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
	U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works — *Voice: (512) 258-0785*
11100 Leafwood Lane — *acw!info@uunet.uu.net* — *BBS: (512) 258-8891*
Austin, Texas 78750-3409 USA — *FAX: (512) 258-1342*

Free surface shipping on prepaid orders For delivery in Texas add 7% MasterCard/VISA

Number 30 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Proposed ANSI C Language Standard

C Ware's DeSmet C V3.1

by
Rex Jaeschke

© Rex Jaeschke, 1988. All rights reserved.

The Standard's almost ready to be voted out, but Rex questions whether conformance is all as he looks at C Ware's latest version of DeSmet C.

Finally, I took delivery of a latest release of a C compiler whose ad doesn't claim that it's ANSI C compatible. It's not that C Ware isn't interested in ANSI conformance (DeSmet C V3.1 shows much evidence to the contrary), but rather that they're more restrained than some vendors. This approach is refreshing. Besides, the DeSmet compiler has a large and loyal band of followers who aren't about to abandon it just because it doesn't have all the latest ANSI goodies. In fact, for the market the compiler sells to it does very nicely, and its speed of compilation and integrated set of tools are among the best in the business. (The SEE editor included in the package has been my only full screen editor on DOS these last five years. It's adequate, powerful, and simple, and I thoroughly recommend it.)

The preprocessor, compiler, and library all come up short in numerous areas of conformance to the proposed ANSI C Standard. However, since C Ware makes no claims to the contrary, this review should not be seen as "negative" simply because some instances of ANSI conformance are missing. Instead, you could look at it from the viewpoint of "here's a long list of new things V3.1 now supports."

The Compiler Options

A couple compiler switches are worth mentioning. **-pt** enables the recognition of ANSI's 3-character trigraph sequences, and **-pw** causes various warning messages to be displayed. Perhaps the most useful of these occurs when you pass an argument to a function and its type is changed by the prototype. No other compiler I have seen does this—I think it's a great idea and shows up lazy programming practices.

No switch exists to make plain **char**s signed.

The Preprocessor

DeSmet C supports the following ANSI C attributes:

- #, ## and **defined** operators
- white space before directives
- **#elif**
- **#include** identifier
- **#error**
- various **pragma**s
- the predefined macros __LINE__, __FILE__, __DATE__ and __TIME__

DeSmet C mimics ANSI's detailed phases of translation quite well, down to the point of allowing any token to be split across lines using the backslash/new-line continuation method.

The predefined macro __STDC__ is indeed defined (and could not be **#undef**ed), and it has the value 0. ANSI C requires this macro to have the value 1 if the compiler conforms to the standard. Since there is not yet a standard, setting the value to 1 would be premature although several compilers already do so. Note, therefore, that you should test __STDC__ specifically against 1 rather than use **#ifdef** __STDC__ since any value other than 1 is permitted for non-conforming implementations.

DeSmet C differs from ANSI C in the following ways:

- The same macro name can be defined differently.
- **#undef**ing a non-existent macro is an error and shouldn't be.
- Non-white space tokens are permitted (and ignored) on directive lines.
- Casts and floating constants are permitted in **#if** expressions, and they shouldn't be.
- The standard headers are not precluded from being **#included** more than once, and they should be.
- While 1024 macros without arguments compiled, only 999 with arguments would, presumably due to the use of the small model.

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ANSI C permits **sizeof** to be rejected by the preprocessor, but it does not require it to be. DeSmet C rejects **sizeof**.

The documentation claims that **#include** files may be nested to a depth of 20. I was able to go only 12 levels despite my CONFIG.SYS file's being set to 30.

The C Language

I found all of the following ANSI features to be supported:

- full function prototypes and new style function definitions
- **signed** keyword
- **const** and **volatile** type qualifiers
- **void** functions
- concatenation of adjacent string literals
- the escape sequences **\xhhh**, **\a**, and **\v**
- the **long double** type
- **long double** and **float** constants
- generic pointers (**void ***)
- **U** suffix on integral types
- initialization of unions
- bit-fields in unions
- **%p** in **printf** for displaying pointers
- *** mask** in **printf** for variable field widths

Although the new keywords **const**, volatile, and **signed** are accepted, they do not appear to have any semantics.

The following ANSI features are not supported:

- initialization of auto arrays and structures
- **long float** is accepted as a synonym for **double** and shouldn't be.
- no **unsigned long int** type

ANSI C Standard Activities

By the time you read this, the September 26th meeting of ANSI C X3J11 should have occurred. The informal consensus among members seems to be that no real obstacles exist that would force us to have another go 'round—provided, of course, that no major holes show up as a result of the latest public review period. If that should indeed be the case, we could well vote out the first ANSI C Standard on Friday, September 30th. In any event, I will provide a report of what actually happens in my next column. In the meantime, fingers crossed—*RJ* ♦

As in numerous compilers I've tested, DeSmet C V3.1 wouldn't recognize **(M)(){}** as a function definition or **int (M)();** as a function declaration. The parentheses around the function name are redundant but legitimate (according to even K&R).

The compiler warns of attempts to define a structure, union, or **enum** template more than once in the same translation unit. However, it does not do so for **typedef**s; that is, the compiler permits you to redefine a **typedef**ed name at any time.

The Runtime Library

Although the library and headers contain almost all of ANSI C's library, numerous macros are missing. ANSI C defines 15 headers. Of these, 5 are missing from DeSmet C and the other 10 are incomplete. The missing headers are: **errno**, **locale**, **signal**, **stddef**, and **time**. As it happens, **errno** and **stddef** can be implemented by the user since no functions are involved. Likewise, **float** and **limits** can be completed by the user. **math** is missing only the hyperbolic functions and **fmod**. **setjmp** seems to be missing only two prototypes.

As far as I could tell, some library facilities are implemented only as macros. ANSI C requires all except those facilities explicitly permitted to be macros to be actual functions. If a safe macro can be written, then and only then are macro versions permitted *as well*.

The Bottom Line

DeSmet C is a ways from conforming to the proposed ANSI C Standard. And although C Ware doesn't claim ANSI conformance or even to be tracking the standard, it is clear they aren't ignoring it either. They have added the lion's share of the new features such as function prototypes and the new preprocessor directives and operators, and in some respects, they are ahead of other compilers claiming to be "up with the standard." Current DeSmet users should be encouraged by this release since it shows that the product is very much alive and growing—despite the intense competition from Borland and Microsoft in the last 1–2 years. ♦

Rex Jaeschke is an independent computer consultant, writer and lecturer. He is also a member of the X3J11 C Standard's Committee, observer on the ISO C Standard's Committee, Founding Editor of the C Journal, *and the C language Editor of* DEC PROFESSIONAL. *Rex's new book* Portability and the C Language, *the first text on this popular topic, will be published by Hayden in September 1988. If you would like to comment on, or question any aspect of, the Proposed ANSI C Standard, write to Rex Jaeschke, 2051 Swans Neck Way, Reston, VA 22091. Rex will endeavor to address your concerns in future columns as space permits.*

FORTRAN TO C

IMPROVED I/O TRANSLATION

FOR_C™

Converts Fortran-77 (with MILSPEC and common extensions) into ANSI C!

• 99% conversion rate on standard Fortran-77 to executable C!
• FOR_C generates C-style prototypes, checking your Fortran for consistent usage, often detecting hidden bugs and suggesting arguments to pass by value in C. By optimizing function calls for C you get clean, concise and accurate translations.
• A C-like preprocessor allows easy customization and added flexibility while translating, e.g. renaming library calls.
• Port code between PC's and mini computers while translating.
• Use FOR_C to maintain both Fortran-77 and C, and for one-time translations.
• C source is included to *all* run-time libraries for maximum portability.
• Source code optimizations produce the most readable and maintainable translation available!

THE FORTRAN TO C TRANSLATOR THAT'S PROVEN, RELIABLE AND FAST!

CALL NOW FOR MORE DETAILS!

$750 MS-DOS DEMO $20

OVERSEAS SHIPPING ADD $30.00

Includes six months' support and free upgrades

XENIX VERSION NOW AVAILABLE

COBALT BLUE

2940 Union Avenue • Suite C • San Jose • CA 95124

Sales (408)723-0474 • Tech.Support (408)723-0495

Number 231 on the Reader Service Card

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A Deal You Can't Refuse ...700 Functions, 19 Disks, Free Software

The largest C Compiler enhancement in the world!

Entelekon's
C BUSINESS PACKAGE
or C STARTER PACKAGE
PLUS

*FREE** **TURBO C®2.0** Borland or *FREE** **QUICK C™** Microsoft or *FREE** **TURBO ASSEMBLER®** Borland or *FREE** Update to Microsoft or Borland products

*OR FREE REFUND if you already own one, see special offer (Limited time offer)

What You Get With Entelekon C Enhancements

GAIN C POWER *Add capabilities your compiler library does NOT have, e.g.:*

- *New! Qwick Menuing—full 1-2-3 like menus & more*
- *Flexible powerful windowing + new Qwick windows*
- *Context sensitive help*
- *Powerful cursor, video and attribute control*
- *Time and date arithmetic*
- *Sample code and working examples*
- *Stacks/queues/linked lists*
- *New! Qwick Data Entry with single function dialog boxes, decision boxes, message boxes*
- *Formatted, fully validated data entry*
- *Display default field values*
- *Calculator style entry option*
- *Full power keyboard handler*
- *700 functions you need*

SAVE TIME, TIME, TIME: *man-years on development, calendar months on schedule!*

SAVE MONEY: *Lowest Cost, Highest Quality Library/Windows Available!*

REDUCE DEBUGGING: *with our thoroughly debugged, bullet proof code!*

SMALLER PROGRAM SIZE: *your application program can be up to 50% smaller!*

EASY *for beginners!* **POWERFUL** *for professionals!*

INSTANT INSTALLATION UTILITY *included!*

SUPERB DOCUMENTATION: *time saving, helpful, clear, complete, instructive.*

FULL SOURCE CODE *included!* **NO ROYALTIES** *on products you develop.*

FREE UTILITY: *To convert Turbo Pascal code to C code.*

LARGEST ADD-ON C LIBRARY *in the world!*

19 DISKS FULL
Source code, documentation, utilities, sample programs
FREE DEMO DISK AVAILABLE

SAVE MONEY! SAVE TIME! DON'T WAIT! ORDER NOW!

SATISFACTION GUARANTEED **(Direct from Entelekon only)** **CALL (713) 468-4412**

POWER WINDOWS™
MOST POWERFUL YET
POP-UP/PULL DOWN/OVERLAP
Menus/Overlays
Messages/Alarms
ZAP ON/OFF SCREEN
FILE-WINDOW MANAGEMENT
Horizontal & Vertical Scrolling
Word Wrap & Line Insertion
Cursor/Attributes/Borders

Full source code $159.95

*SPECIAL OFFER
Free Turbo C, QuickC or Turbo Assembler with purchase of C Starter Package or C Business Library. Even if you already own Turbo C, QuickC or Turbo Debugger, we will refund up to $100.00 with the purchase of C Starter Package or C Business Library.

C FUNCTION LIBRARY
BEST YOU CAN GET
OVER 500 FUNCTIONS
FULLY TESTED
BETTER FUNCTIONS

Full source code $159.95

C BUSINESS LIBRARY
INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C, B-TREE LIBRARY, ISAM

ALL for $299.95
(A $500.00 VALUE)

B-TREE LIBRARY & ISAM DRIVER
POWERFUL DATA MANAGER
FAST! EASY TO USE!
16.7 MILLION RECORDS/FILE
16.7 MILLION KEYS/FILE
Fixed/Variable length records.

Full source. No royalties $129.95
Multi-User option available

C STARTER PACKAGE
INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C

ALL for $199.95
(A $370.00 VALUE)

Entelekon™
SINCE 1982

12118 Kimberley, Houston, TX 77024 713-468-4412 VISA-MASTERCARD-CHECK-COD

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A Forth Apologia

by
Ray Duncan

I have yet to see a decent piece of software written in Forth. Let's face it. Forth stinks. —John Dvorak, provocateur and columnist, *InfoWorld*, October 29, 1984

Forth is the first language which has been honed against the rock of experience before being cast into bronze. —Charles Moore, inventor of Forth

Only brain-damaged programmers use Forth. —Allen Holub, ex-C columnist, *Dr. Dobb's Journal*

Forth is like the T'ao; it is a Way, and is realized when followed. Its fragility is its strength; its simplicity is its direction. —Michael Ham, ex-Forth columnist, *Dr. Dobb's Journal*

As the pearls of wisdom above demonstrate, the programming language Forth does not engender lukewarm sentiments in the hearts of its beholders. With rare exceptions, Forth programmers are passionately enthusiastic about the language and will bend your ear for hours about how it changed their lives. Most other programmers view Forth with the distaste mingled with wary respect ordinarily reserved for Mafia chieftains and the authors of software viruses.

How did Forth acquire this reputation as the "bad boy" of programming languages? Some of its reputation is undoubtedly based on hearsay passed on by pundits like Mr. Dvorak and Mr. Pournelle, on *Rolling Stone* interviews with scruffy-looking Northern California hackers, on well-meaning but misguided promotional efforts by Forth groupies, and on the chaotic appearing, uncommented Forth source code published in trade magazines (when any code is published at all). Some of Forth's bad reputation stems from sensational (and sensationalized) disasters like EasyWriter and Valdocs, both of which happened to be written in Forth. In addition, there is always the snobbery factor and the Not-Invented-Here syndrome; "if they didn't teach me about it at Stanford in Computing Science 1A, it can't possibly be any good."

But the largest ingredient in Forth's roguish renown seems to be a simple lack of information. In the course of my own work, which involves approximately equal amounts of assembly, C, and Forth programming, I have been repeatedly astonished and appalled by the degree of ignorance about Forth in the general programming community. Microsoft, for example, recently billed the use of threaded code in its QuickBASIC 4.0 product as a technological breakthrough, when exactly the same technique has been used in Forth systems for nearly twenty years. Reviewers marvel over the new integrated programming environments from Borland and Microsoft, oblivious to the fact that Forth was providing integrated environments with an interpreter, compiler, editor, and assembler in fewer than 16 KB of RAM for nearly 15 years before Turbo Pascal. Many of the authoritative reference works on programming languages don't mention Forth at all.

I find the contemptuous attitude toward Forth so common among C programmers especially disconcerting because the two languages are virtually contemporary, were designed with many of the same objectives, and require much the same sort of approach when solving a particular programming problem. This impression of similarity is supported, in my experience, by the ease with which Forth programs can be translated to C and vice versa. Applications written in Forth and C are also prone to many of the same sorts of bugs because both languages are tightly coupled to the hardware and allow you to mangle pointers and data in any way you like.

Forth is, like every other programming language, simply a tool with particular strengths and weaknesses that reflect the original concerns of its designer. In the appropriate settings, Forth is without peer, just as FORTRAN is inarguably the best tool for crunching numbers, LISP for crunching text, and Prolog for crunching rules. While it is not my intention to teach Forth programming in this article, I would like to place some of the claims and counterclaims about Forth in a more reasonable perspective. A little consciousness-raising never hurt anyone, and if at some point in the future you recognize a job that is especially suited to Forth and don't just bulldoze your way through with your "usual" language, you might save yourself weeks or months of work.

Some Forth History

Although Forth and C were conceived at almost the same time, their origins are vastly different. C, with its fraternal twin Unix, was born in a very bastion of computer science academia—Bell Labs—and its author Dennis Ritchie has ascended in the ensuing years to near-sainthood status. For a long time, quality C compilers were available only in the Unix environment, and while Unix was fiendishly expensive to license for commercial users, AT&T practically gave it away to universities. Thus, an entire generation of computer science students and faculty were weaned and raised on Unix with C and made that their true faith.

Forth is a grassroots language; it has made its way in the world strictly on its own merits, lacking endorsements from the learned professors or distinguished looking textbooks from Addison-Wesley or Springer-Verlag. Forth was invented by Charles Moore, a freelance programmer who had worked on control and data acquisition applications in environments ranging from a carpet factory to the Stanford Linear Accelerator Center. Moore was familiar with many programming languages, including assembly, FORTRAN, COBOL, PL/1, and APL, but felt that none of them met his needs well. He experimented extensively with interpreters in the early 1960s, eventually hitting on the idea of using a stack to pass parameters and results between the interpreted commands.

As Moore recounts the story, his first interpreter that was recognizably Forth was written in FORTRAN on an IBM 1130 while he was working at a carpet manufacturer (Mohasco Industries) in 1968. He subsequently cross-compiled the inter-

PRODUCED 2009 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

preter to a Burroughs 5500, which was a CPU with push-down stacks that supported Forth much more gracefully, and then to a Univac 1108. Shortly afterward, Moore went to work for the National Radio Astronomy Observatory (NRAO) and ported Forth to a Honeywell 316. The language had been evolving rapidly during these ports, and a compiler was added in the Honeywell version, resulting in the first Forth system in the modern sense and making it possible (eventually) to rewrite Forth in itself.

NRAO didn't appreciate the implications of Forth and after some preliminary investigation into its patentability released all rights to the language to Moore. Astronomers at Kitt Peak Observatory were soon using Forth on several Varian minicomputers for telescope control and data acquisition. In 1973, Moore, Elizabeth Rather (the second Forth programmer), and several others formed FORTH Inc. to sell Forth as a commercial product, originally called miniFORTH and later polyFORTH. The language was soon running on several additional processors; the most important of these was the DEC PDP-11, which remains a mainstay of the FORTH Inc. user base today.

For the next five years, Forth the language and Forth the proprietary product of FORTH Inc. were one and the same. There were no other commercial vendors of Forth systems, and in fact the name Forth itself was trademarked. Although FORTH Inc. moved polyFORTH onto a broad variety of microprocessors as they became available, it maintained prices at mainframe levels. Except for a few scattered systems (descended from Moore's original NRAO implementation) in use at university astronomy departments, Forth was essentially inaccessible to casual or tentative users.

In 1978, a small band of Forth enthusiasts formed the Forth Interest Group (FIG) in San Carlos, California. They assembled a team that over a nine month period designed a simple, transportable implementation of the Forth language and then implemented it on the Intel 8080, the DEC PDP-11, the TI 9900, the 6502, the PACE, and the Motorola 6800. This implementation, called FIG-Forth, was released into the public domain; xeroxed listings were sold for a nominal fee ($10) and disseminated worldwide in a matter of months. Some of the purchasers of the original listings performed further ports to the Intel 8086, the Nova, the RCA 1802, the Motorola 68000 and 6809, the Alpha Micro, the VAX, and even the Data General Eclipse, which were also placed in the public domain.

Although the FIG-Forth implementations were slow and crude by today's standards, they were perfectly suited to the times. In those days, RAM was dear, 5.25" floppy disks held 60–70 KB, and cassette tape or even paper tape was still a common mass storage device. A high-level interactive language with minimal memory requirements, available with complete source code, was the answer to many microcomputer experimenters' prayers. The FIG-Forth movement was given an additional impetus by the August 1980, issue of *BYTE*, which was devoted almost entirely to Forth articles and exposed Forth concepts to the general microcomputing community for the first time.

The FIG-Forth listings also provided a launching pad for dozens of would-be Forth entrepreneurs. Since the listings reduced the ante for becoming a programming language vendor to practically zero, Forth software houses sprang up by the dozens, several for each microcomputer that was popular at the time. Although most of these Forth-vendor-wanna-be's did not survive, the influx of new blood brought about rapid improvements in Forth development tools. For example, prior to 1979, stand-alone Forth systems with line editors and integer arithmetic only were the rule. By 1981, Forth systems that ran as well-behaved tasks under a host operating system, used normal files for program and data storage, and supported floating point arithmetic and visual editors were readily available from the software houses that had gotten their start with FIG-Forth.

With the success of the FIG-Forth implementations under their belts, the founders of the Forth Interest Group looked for new fish to fry. The Forth Standards Team, a self-appointed spin-off from FIG, commenced deliberations to "improve" the language and issued the Forth-79 Standard in October 1980. Forth-79 was incompatible with both FIG-Forth and polyFORTH, failed to supplant either, and consequently had a most regrettable effect on Forth source code portability and the publishing of Forth textbooks and articles. Although it may seem difficult to believe, the Forth Standards Team proceeded to splinter the Forth community even further with the release of the Forth-83 Standard, which was (again) incompatible with all existing Forth dialects including Forth-79.

Forth-83, with all its problems, did have two highly desirable side-effects. First, the public domain FIG-Forth listings, which by 1983 were severely dated, were replaced virtually overnight by a highly popular public domain implementation of Forth-83 called F83, written by Henry Laxen and Michael Perry. F83 can recompile itself (FIG-Forth required an external assembler), has multitasking capabilities, runs under the host operating system, and generally represents a quantum jump in power and sophistication over FIG-Forth. Second, rumblings in 1987 of a reactivation of the Forth Standards Team caused major Forth vendors and users to band together and launch an ANSI standardization effort. The

1988 Real-Time Programming Convention

Sponsored by the Forth Interest Group

The 1988 Real-Time Programming Convention will be held November 18–19 at the Grand Hotel in Anaheim, California. Jef Raskin, head of the original Macintosh development team and inventor of the Canon Cat, and Ray Duncan are the invited speakers.

The focus of the convention will be on software solutions to real-time programming problems. Talks and demonstrations will cover languages and operating systems for real-time data acquisition and process control, language-oriented RISC machines, high-performance embedded systems, intelligent instrumentation, machine vision, robotics, and more. Charles Moore, inventor of the Forth language, will be present. For more information, contact Kent Stafford, Forth Interest Group, 1330 S. Bascom Avenue #D, San Jose, CA 95128.

A $1,000 prize will be offered to the programmer who puts FIG's "mystery gizmo" through its paces the fastest, using any computer, any software. The showdown will be at the Real-Time Programming Convention. For more information about the contest, contact the Forth Interest Group, P.O. Box 8231, San Jose, CA 95155.♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

X3J14 Technical Committee, which is drafting an ANSI Forth Standard under the auspices of both CBEMA and the IEEE Computer Society, appears to have drawn the fangs of the Forth Standards Team forever.

In the last few years, the Forth language vendors have undergone a severe shake-out, and only a half-dozen with solid product lines and significant user bases remain. (Judging by the number of companies managing to make a living in the much larger C marketplace, this may still be too many for the long haul.) FORTH Inc., with Elizabeth Rather as president, is clearly the flagship of the vendors; its revenues and staff exceed those of all the other Forth software houses taken together. Only FORTH Inc. has achieved the critical mass to offer a complete range of services: consulting, custom applications programming, education, and tools; the other vendors appear to subsist almost entirely on sales of packaged development systems.

While the pace of change in Forth programming tools has slowed, some exciting things have been happening on the hardware front. Several CPUs have been designed and built with Forth instruction sets, including Charles Moore's Novix NC 4016, Alan Winfield's MF1600, and Phil Koopman's WISC (Writable Instruction Set Computer) CPU/32. The performance of these machines is quite astonishing. The Novix processor, for example, can crank out well over 4 MIPS, yet requires so little power and so few support chips that it can easily be built into a handheld computer.

C	Forth
break	LEAVE (loops only)
do *statement(s)* while (*expression*)	BEGIN *statement(s) expression* WHILE *statement(s)* REPEAT
for (*expr;expr;expr*) *statement(s)*	*limit start* DO *statement(s)* LOOP or *limit start* DO *statement(s) n* +LOOP
goto *name*	*(no equivalent)*
if *(expression) statement(s)* else *statement(s)*	*expression* IF *statement(s)* ELSE *statement(s)* THEN
return (*expression*)	EXIT

switch (*expression*) {
case *n*: *statement(s)*
...
default: *statement(s)*
}
	expression CASE
	n OF *statement(s)* ENDOF
	...
	default statements
	ENDCASE

while (*expression*) *statement(s)*	BEGIN *statement(s)* *expression* UNTIL

Figure 1. Structured control constructs in C and Forth. The Forth CASE statement shown is not standardized but is in common use.

Some Forth Terminology

Depending on whom you talk to—or believe—Forth is a language, an environment, an operating system, and a philosophy of programming. All these claims are true, in some sense, but this semantic overload of the term Forth sometimes gives "outsiders" trying to understand what Forth is about the sensation of being smothered in cotton wool. It doesn't help matters that Forth programmers use a unique—and somewhat bizarre—terminology when discussing how the language works and is structured.

Every Forth language element that can be used interactively or within a program is called a *word*. (The term *word* is also severely overloaded in Forth literature; in some contexts it can mean merely "a blank-delimited token from the input stream," and there is also a parsing operator named *WORD*.) The source code for a particular word is called its *definition*. Some words are *primitives*, coded in the CPU's native machine language; the remainder are *high level* or *secondary* definitions, coded in Forth itself.

The Forth interpreter/compiler looks words up in the *dictionary*, which is essentially a symbol table that associates each word's name with the address of the corresponding executable code. The dictionary can be subdivided into *vocabularies* that can be searched independently or in a tiered fashion; this allows names to be overloaded and also allows "dangerous" or infrequently used words to be "hidden" so that they are not invoked accidentally.

Why do we concern ourselves with this strange nomenclature at all? For the purely pragmatic reason that to communicate with a Forth programmer, you need to use his terminology, because he probably won't understand yours. Terms like *word*, *dictionary*, *vocabulary*, and *definition* have the force of "tradition" behind them and pervade every Forth book, article, and set of source code in existence. Furthermore, many Forth programmers are self-taught, learned Forth only as a means to accomplish something else, and have never used any other programming language. Backgrounds in electrical or mechanical engineering, medicine, natural languages, music, astronomy, and the like are common; formal educations in computer science or mathematics are rare.

The Forth Language

Viewed strictly as a language, Forth is clearly in the camp of the modern, structured languages such as C, Pascal, and Modula-2. Forth has all the proper flow of control constructs and, unlike C, does *not* have a GOTO (Figure 1). Like C, but unlike Pascal and Modula-2, Forth draws no distinction between procedures and functions. A more important difference between Forth and the other structured languages is that Forth has no support for data typing either at compile time or at runtime.

The key difference between Forth and the other structured languages is that the Forth compiler is *extensible*. Forth allows the programmer to declare new data types *and* to define new compiler keywords (such as control structures) and then use them immediately—even within the very next procedure. Forth even has a compiler facility for creating new kinds of language objects, if you will: The programmer can separately define what happens at compile time (when the object is instantiated) and what happens at runtime (when the object is invoked). The closest C comes to being extensible is to allow you to build record formats out of existing data types with **struct**.

But Forth is not simply a set of keywords and syntax rules; it is a description of a virtual machine. The most important aspect of the virtual machine is that it has exactly two stacks: a parameter stack and a return stack. Nearly all Forth words use the parameter stack for both their arguments and their results; the return stack is used for flow of control within a program and (occasionally) for temporary storage of working values. Forth lends itself naturally to recursive algorithms be-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

cause of this stack-based architecture; in "classic" Forth programming, global variables are frowned on and rarely used.

The extant Forth standards specify the names and actions of fewer than 200 words, which suffice to define the Forth virtual machine and the behavior of the interpreter/compiler. Commercial Forth development systems are much more elaborate, and have at least 500 words resident and available in the interpreter/compiler. Unfortunately, in many Forth manuals the words are simply documented in ASCII collating sequence and are discussed as functional groups only as an afterthought (if at all). This makes the prospect of mastering a Forth system quite forbidding.

The X3J14 Technical Committee appears to have drawn the fangs of the Forth Standards Team forever.

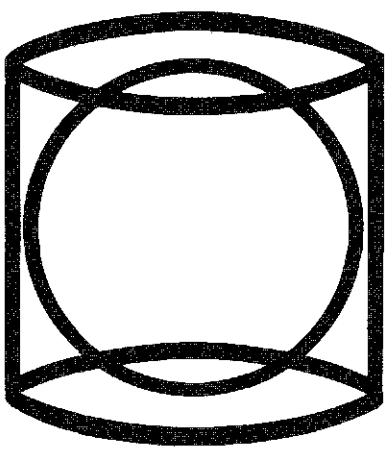
Actually, 500–600 language elements is almost exactly the same as the combined number of operators, keywords, and runtime library functions in Microsoft C 5.1 or Turbo C 1.5, and you can study Forth *initially* with much the same approach you would use for C. As a first approximation, you can view the relatively few "magic" words known to the Forth compiler as the "language" and treat the rest as a runtime library divided into functional categories such as stack operators, arithmetic/logical operators, memory access operators, and so on. As with any language, you will use 10% of the library 90% of the time, so you can learn just the most common functions first, and worry about the rest when you need them.

Once you've got a basic grasp of what's available in the Forth language, it's better to use a "layered" model to think about the system because this is the way the system is actually built up. The lowest layer consists of the primitives, which are written in the assembly language of the host CPU and implement the Forth virtual machine. The words coded as primitives tend to be the most frequently used arithmetic/logical, comparison, stack, and memory operators along with a few text-parsing building blocks for the interpreter/compiler. Each additional layer contains increasingly complex functions built out of the words defined in the layers below: console I/O, mass storage, formatting and string management, the interpreter, the compiler, and at the top, utilities such as the editor and assembler.

Two aspects of Forth that are frequently criticized are its cryptic names and its postfix (or *reverse Polish*) syntax. To address the former: When the fundamental Forth names were assigned, 10 character-per-second Teletypes were common and so were minicomputer systems with 8 or 16 KB of RAM. Consequently, short names were highly desirable to conserve memory and keystrokes, and we ended up with symbols such as @ for a memory fetch, ! for a memory store, and so on. I won't make any attempt to defend the historical Forth namings; each language has its conventions, and Forth's are no stranger than some found in C, LISP, or APL. Readable, maintainable programs can be written in any language, as can write-only, unmaintainable programs. The secret of obtaining the former rather than the latter is good design, discipline, and documen-

HS/FORTH

When only the best will do:
Fastest Execution
Fastest Compilation
Most Functionality
Most Flexibility
Best DOS Interface
Best Architecture
And a 30 day guarantee!



HS/FORTH SYSTEMS
STUDENT **$145.**
PERSONAL **$245.**
software float, quad integer, optimizer
PROFESSIONAL **$395.**
hardware float, turnkey, assembler, screens, RSDL linker, round robin & time slice multitasker, dynamic string manager
PRODUCTION **$495.**
DOS & ROM metacompilers, system ROMer, Meta-Windows data structures & objects
GIGAFORTH **$295.**
Native mode for AT

HARVARD SOFTWORKS
NUMBER ONE IN FORTH INNOVATION

(513) 748-0390 P.O. Box 69, Springboro, OH 45066

Number **42** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tation, not use of a particular language. At least in Forth, since it is extensible, you can rename any language element to anything you like!

The complaints about Forth's postfix syntax are more to the point, and deserve a more cogent response. In postfix systems, the arguments precede the operator; for example, to add 1 and 2 in Forth you would write,

```
1 2 +
```

whereas in C, which uses infix syntax, you would code,

```
1 + 2
```

and in LISP, which employs prefix syntax,

```
+ 1 2
```

When recursive descent compilers for C and Pascal parse your pretty infix expressions, they transform the expressions into an intermediate postfix form before generating machine code. Forth is postfix to start with to shift the burden of expression evaluation onto the programmer so that the compiler can be simpler, smaller, and faster. While postfix notation certainly takes some getting used to, the success of HP calculators indicates that postfix should not be considered a major barrier to learning Forth. It is interesting to note that (because the Forth compiler is extensible) an infix expression evaluator can easily be layered onto the compiler; several have been published, but Forth programmers have apparently not found them necessary or useful.

The Forth Environment

The C, Pascal, and Modula-2 languages are typically implemented as native code compilers, and true interpreters for these languages (when they exist at all) are usually slow and cumbersome. Although the so-called integrated environments for these languages, beginning with Turbo Pascal in 1983, have improved programmer productivity immensely, the environments are really just co-resident editors, compilers, and linkers with a shell that allows the programmer to jump quickly from one to the other. They are not interpretive environments: You cannot invoke any language element, or a procedure you have written yourself, interactively by typing its name together with some parameters and look at the results.

Forth, on the other hand, is the original interpretive, interactive, integrated environment. In typical systems, the editor, assembler, and various debugging utilities are written in Forth itself and are kept resident at all times. The source code for these tools is usually kept on-line so that the programmer can tune them to personal taste.

Although Forth has been implemented in many ways, the "classic" implementation is a unified interpreter and incremental compiler based on some startlingly simple concepts and ground rules. The core of the interpreter/compiler is a loop that reads blank-delimited tokens from the input stream (which can be either the keyboard or blocks of text from mass storage); the disposition of each token is controlled by the global variable STATE.

When STATE is FALSE, the system is said to be "interpreting." Each token is looked up in the dictionary; if found, the corresponding code is executed. If the token is not found, the interpreter attempts to convert it into a number, which is left on the parameter stack; if the conversion is unsuccessful, the interpreter issues an error message, clears the stack, and waits for a new command.

If STATE is TRUE, the system is "compiling" a new definition. If the token is found in the dictionary, some representation of the word (usually just a pointer) is compiled for later execution. If the token is not found but can be converted to a number, code is generated which will push that value onto the parameter stack when the new definition is executed. In other words, each "compiled" Forth definition is just a list of addresses and literal data, called "threaded code." The addresses refer, directly or indirectly, to executable machine code. When threaded code is "executed," a short routine called the "inner interpreter" walks down the lists of addresses and runs the corresponding machine language procedures.

As soon as a definition has been compiled, it can be interpreted—executed interactively by typing its name—or it can be referenced in subsequent definitions. All definitions in a Forth system have equal status, and there is no distinction or artificial barrier between the definitions that constitute the lower layers of the system (including the interpreter/compiler) and those added later as part of an application. An entire Forth program is run by just entering the name of its "highest definition."

Two special classes of words make the Forth compiler work: *defining words* and *immediate words*. Defining words create a new entry in the dictionary—that is, add a new symbol to the symbol table—and trigger the construction of a new variable, constant, procedure, or other object. The defining word that begins a new procedure—the colon (:)—changes STATE from FALSE to TRUE and thus initiates compilation. The application programmer creates new classes of Forth language objects by defining new defining words, and creates new members of the classes by invoking the new defining words.

Immediate words are so called because they are always executed immediately regardless of the value of STATE. The most important immediate words are the control structure words and the semicolon (;). The control structure words (such as IF or BEGIN) perform syntax checking and then generate appropriate code; for example, IF generates code that tests the value on top of the parameter stack and then performs a conditional branch on the result. The semicolon (;) is the terminator for a definition previously begun with the colon (:); its main action is to change the value of STATE back to FALSE. The application programmer extends the Forth compiler by defining new immediate words.

The Forth Operating System

For the first ten years of Forth's existence, it was always implemented and sold as a "native" system; that is, not only was it a programming environment, but it was the *whole* environment. Native Forth systems contain their own device drivers and do not run under the control of a host operating system; they take over the entire machine. Moore's goals were speed and compactness at any price, and he was perfectly willing to give up the benefits of hardware independence and file systems in return.

Forth interpreter/compilers that ran under host operating systems did not appear until 1979, in the first wave of small software houses that appeared at the time of the FIG-Forth listings. Two of the pioneers in this area were Laboratory Microsystems Inc. (my own company), which sold a Z-80 Forth for CP/M that used operating system calls for I/O and file management, and Micromotion, which created a similar

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

system for the Apple II. Over time, the advantages of host OS-based Forth interpreter/compilers became obvious, and today about 80% of the Forth programming systems sold fall into this category. So far as I am aware, FORTH Inc. is the only company still selling native Forth systems as a packaged product, but it now also produces systems that run under MS-DOS, RSX-11, and VAX VMS (among others).

In embedded applications, of course, Forth continues to be used in its native incarnation to reduce hardware requirements to a minimum. A special type of Forth compiler, called a target compiler, runs on a full-fledged development system and creates a stand-alone executable image (which can be ROMable). The image is then transferred to the target system by any convenient means—ROM, EPROM, downloading over a serial link, or diskette. Naturally, a target compiler can also be a cross compiler; since the target compiler itself is written in Forth, the CPU upon which the executable image is compiled need not be the same as the CPU upon which the image will be executed.

The Forth Philosophy

If Forth has a credo, it might be summed up as smallness, simplicity, and speed. Forth programmers traditionally disdain the flashy user interfaces and elaborate error handling that characterize most integrated environments nowadays. This attitude has its benefits—for example, a self-contained interactive Forth system including an editor, assembler, and even multitasking support can easily be put in an 8 KB EPROM. It also has its drawbacks—runtime error checking in a Forth system is virtually absent (unless it happens to come "for free," as in divide-by-zero detection on the Intel 80x86 family). Frequent total system crashes during Forth program development are taken for granted; indeed, they are expected.

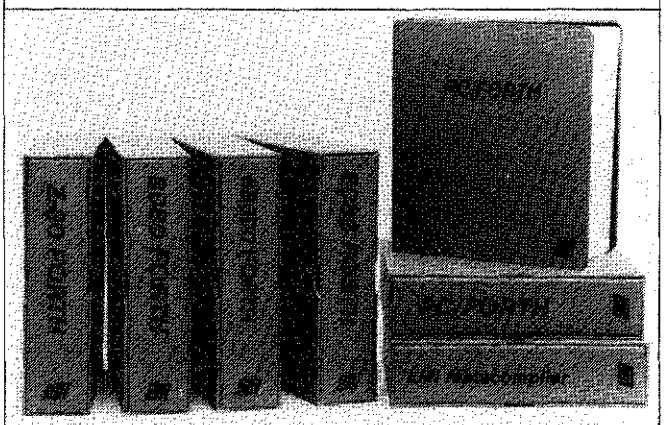
Corollaries to the Forth credo are that intimate access to the hardware should always be available and portability should never get in the way of exploiting a CPU's instruction set to its fullest. A Forth system without an assembler is felt by most to be only slightly better than no Forth system at all. As for portability, you would be amazed at the number of arguments in the ANSI Forth Technical Committee that have hinged on whether a particular language feature can be implemented efficiently on someone's favorite processor.

Classic Forth programming style is characterized by many, many short, relatively simple definitions, bottom-up design, prototyping, and successive approximations. Forth dogma has it that an ideal definition should not exceed 2 or 3 lines and should not contain more than 9 or 10 elements. This ensures that individual definitions can be exhaustively tested; the programmer can simply put parameters on the stack and exercise the definition interactively through all of its possible internal paths. It also constrains an individual definition from becoming too specific so that it is more likely to be reusable in the application.

The bottom-up design of a Forth application derives naturally from the types of problems to which the language is routinely applied. A typical Forth application involves intimate interaction with hardware to acquire data or control a machine; in many cases, the hardware itself is only poorly understood or is not yet stable. The Forth words that are most hardware dependent are thus usually coded first; when these definitions are proved and the hardware is well understood, the rest of the application is built up by adding layers of increasing sophistication and complexity.

Continued . . .

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number **63** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The emphasis on prototyping and successive approximations has much the same grounds as the stress on bottom-up design. In most Forth applications, getting the hardware to do what you want puts you a long ways towards your goal. After prototyping some code that approximates the final form of the application, you can simply discard or modify the upper layers until you get the specific functionality and user interface that you need; the lower, more hardware dependent levels of the application rarely require any significant change.

An important advantage of Forth in application design is that the documented user interface can rely on Forth's own interpreter and merely consist of the top layer of definitions. This offers the sophisticated user incredible flexibility, because not only can he use the documented application entry points, but he can extend the application by putting building blocks together in new ways, or access lower level words to test individual hardware components. In environments with naive users, the Forth compiler and the interpretation of "dangerous" Forth words are disabled, or an application-specific, self-contained user interface is written that does not lean on the Forth interpreter.

Forth's Place in the Universe

In 1969 there was exactly one Forth programmer: Charles Moore. When FORTH Inc. was founded in 1973, there were still probably fewer than a dozen. In 1979, Moore estimated that there were about 1000 Forth programmers. In 1988, based on the number of packaged systems that have been shipped by the larger vendors and the copies in print of Leo Brodie's *Starting Forth* (a popular Forth primer), I'd conjecture that there are as many as 50,000 serious Forth programmers and another 25,000–50,000 casual users. This is certainly respectable growth for a grassroots language. On the other hand, just to keep things in their proper perspective, something like 1.5 million copies of Turbo Pascal have been sold to date, never mind C compilers.

Looking back over Forth's somewhat checkered career, I can identify four distinct phases. I think of the first, spanning the years 1969 to 1978, as the "secret weapon" era. During this time, Forth evolved rapidly as it was used by a small set of people on highly specialized, demanding projects. There was no significant user base whose code would become obsolete when the language was changed, so many new concepts could be tried out and added to the language whenever they proved generally useful.

The second era, 1978 through 1982, marked the transition of Forth from a little-known proprietary product to a general purpose programming language. We could call this the "evangelical era" since it was marked by an intense proselytizing effort by FIG, augmented by Forth's position as the first inexpensive structured language with graphics, floating point, and its own editor and assembler on the immensely popular IBM PC (and later on the Macintosh). In retrospect, we owe the FIG founders a tremendous debt for legitimizing the language by seeding the "second-sources." Fortunately for us all, FORTH Inc. had the foresight not to enforce its trademark on the name "Forth"; I trust they have benefited more than they have suffered by this policy.

The years 1983 through 1986 were a period of Forth retrenchment and disillusionment. Most of the cultists, hobbyists, and nut cases drifted away from Forth as its novelty faded, and many of the professionals who continued to use Forth felt frustrated by its lack of penetration of the computer mainstream. They were convinced that they had a better solution for many classes of programming problems, but most computer science departments and the larger trade magazines continued to ignore Forth's existence. Smaller Forth vendors were driven out of business by advertising costs, the effort required to keep up with rapidly changing PC hardware, public domain Forths that could be downloaded from bulletin board systems, and the vendors' tendency to compete among themselves instead of trying to broaden the market.

In 1987, we entered the era of Forth maturation and true standardization. The ANSI Forth Technical Committee has been making excellent progress in reconciling the existing Forth dialects, and we hope to see a draft proposed ANSI Forth Standard released for public comment by the end of 1989. While an ANSI Forth Standard will not be an instant cure for all Forth's ills, it should drastically improve the portability of both Forth applications and Forth programmers, and thus make Forth a more viable candidate for consideration by project planners.

Forth development systems are available for nearly every CPU in existence, and Forth is consistently the first third-party high level language to appear on a new CPU architecture, personal computer, or operating system. Recent examples of this are the Intel 8096/97 microcontroller, the TI TMS34010 graphics coprocessor, the TI TMS32020 digital signal processor, NASA's Massively Parallel Processor project, and Microsoft's OS/2 operating system. Another interesting development (although its implications for Forth as a language are still unclear) is that the Novix NC 4016 design has been licensed by Harris Semiconductor and integrated into their cell library and is being aggressively marketed as the Harris RTX-2000 for applications requiring an inexpensive, high-performance microcontroller.

Forth is enjoying increasingly wide use in those real time control and data acquisition applications to which it is best suited and can be found in the most unexpected places: ranging from the bar-code-reader "wand" carried by your friendly Federal Express driver, to the American Airlines baggage control system, to instrumentation for SpaceLab experiments, to a networked monitoring and control system with some 400 processors and 17,000 sensors at an airport in Saudi Arabia. While I think it likely that Forth will always remain a somewhat obscure language, its speed and interactivity coupled with minimal hardware requirements and ready portability are a unique combination of assets. In its niche, Forth has no real competition, and I expect it will be around for a long time to come.

Further Reading

Brodie, Leo. *Starting Forth*. 2d. ed. Englewood Cliffs, N.J.: Prentice Hall, 1987. ISBN 0-13-843079-9.

Brodie, Leo. *Thinking Forth*. Englewood Cliffs, N.J.: Prentice Hall, 1984. ISBN 0-13-917568-7.

Moore, C.H. "FORTH: A New Way to Program a Mini-Computer." *Astron. Astrophys. Suppl.* 15 (1974): 497-511.

Information about the ANSI Forth Standard effort can be obtained from the X3J14 Secretary, 111 N. Sepulveda Blvd., Suite 300, Manhattan Beach, CA 90266. ♦

Ray Duncan, author of the MS-DOS "bible" Advanced MS-DOS *and General Editor of the* MS-DOS Encyclopedia *(both published by Microsoft Press), is president of Laboratory Microsystems Inc. LMI sells Forth interpreters, compilers, and cross-compilers for a variety of microprocessors and does consulting work in assembler, Forth, and C. Ray Duncan's next book* Advanced OS/2 *(also from Microsoft Press) will be published in the first quarter of 1989.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

QEdit *Advanced*

the "quick editor"

By SEMWARE

QEdit is one of the smallest, fastest, most easy to use multi-file editors available for IBM Compatible Computers with many features to enhance your productivity.

- ***Extremely Fast***
- ***Completely Configurable (Including Keyboard!)***
- ***Easy to Use Macro Facility***
- ***Optional "Pop Down" Menu System***
- ***Recover Deleted Text***
- ***Edit Dozens of Files Simultaneously***
- ***Utilize up to 8 Windows to View Files***

QEdit *Advanced* requires an IBM PC/XT/AT or Compatible, IBM PS/2 30-80, 128KB of available memory. Please specify 5.25″ or 3.5″ diskette.

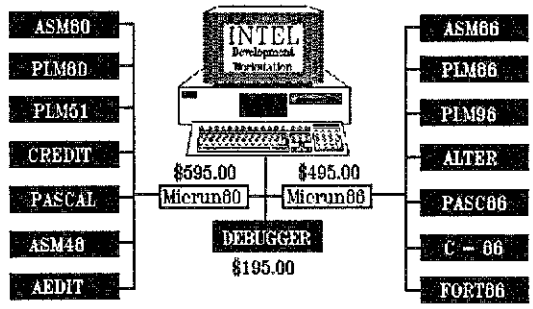
$54.95 MASTERCARD & VISA ACCEPTED

Add $3.00 Shipping & Handling. Georgia Residents Include 4% Sales Tax.

To Order:
SEMWARE, 730 Elk Cove Court
Kennesaw, Georgia 30144
Phone Orders: (404) 428-6416

Number 148 on the Reader Service Card

RUN INTEL ON A PC



DEVELOP and DEBUG

8 and 16 Bit INTEL CODE ON **YOUR** PC...WITH

MICRUN

MICRUN 80....8 Bit Development Tool (6 or 10 MHZ)
MICRUN 86.....16 Bit Development Tool
8080 Debugger.....Intel Type Software Debugger

- **Micrun 80/86 Package $ 895**
  (includes Micrun 80,86 & 8080 Debugger)

****

INTRODUCING
THE PC/INTEL DEVELOPMENT STATION

INCLUDES:

- **Floppy Drive w/Controller**
- **MICRUN 80/86 PACKAGE**
- **MS-DOS & GW BASIC**
- **QUIKSAVE-Backup Utility Program**

XT 10 MHZ V-20	AT 12 MHZ 80286
$ 2,195	**$ 2,995**

All systems ready to go. Pretested (48 hr burn-in). Optional equipment availble.

Call Today **1-800-888-8086**

Micro Interfaces Corp. 16359 NW 57th Ave., Miami, FL. 33014

*Intel, Everex & Seagate are registered trademarks of their respective companies.

Number 214 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programming for Reliability

Data Types and Exceptions in Ada

by
Richard D. Riehle

There are some trees, Watson, that grow to a certain height and then develop some eccentricity.

—Sherlock Holmes

It is 1995, and you are wearing the pacemaker installed two years ago when you were diagnosed with chronic arrhythmia due to a sino-atrial block. Your pacemaker is one of a new generation of such devices controlled by a microcomputer with embedded software: a real-time system (RTS). If the software should "ABEND," so might you.

In our last article we discussed some goals of software engineering—Reliability, Efficiency, Modifiability, Understandability, Portability.

In this installment, we explore some Ada ideas for making a system consistently do what it is supposed to do. The reliability issues are fault tolerance, correct operation, and acceptable precision of results.

In the hardware world we consider reliability in terms of Mean Time Before Failure (MTBF) and Mean Time To Repair (MTTR). And maintenance is viewed as predictive, preventive and corrective. In "mission critical" real-time systems software, prediction, prevention, and correction must be an integral part of the total operating environment. RTS software must perform reliably forever or until the death of its mission, whichever comes first. Truly effective RTS software may even perform its own predictive, preventive, and corrective maintenance thereby delaying the death of its mission.

In "The Uniqueness of Man," Dr. Julian Huxley emphasized the capacity of human beings to be both brilliant and stupid, sane and insane. Mental flexibility is a trait with which we alone, as intelligent creatures, are endowed. And that very flexibility is the source of both our reliability and our unreliability. *Cogito ergo sum* is a blessing and a curse.

Maintenance Definitions

predictive maintenance	=	based on some knowledge of the characteristics of the the environment, replace or adjust parts to compensate for a predicted probable failure rate
preventive maintenance	=	do some kind of maintenance on a pre-set schedule, based on our best guess
corrective maintenance	=	when it breaks, fix it

Software systems often inherit our curse. The more flexibility they have, the greater the chance for error. Conversely, we can use that flexibility to create almost any kind of simulated world we can imagine. In software, the most flexible development tool is, of course, raw machine language. Next is Assembler language. The more access the language provides to low-level computing facilities, the greater the flexibility and the higher the probability for error. A language such as C is enormously flexible, but it's not a good choice for writing large, reliable software systems.

Somewhere in the computer language hierarchy we need software development tools that permit us to improve the chances of reliability without sacrificing creativity. Ada is one of these tools.

An Overview

One approach to achieving reliable systems is to use fully tested components that already exist. Another trick is to encapsulate data descriptions and procedures in small, independently verifiable components. Ada permits both of these methods via packages, abstract data types, and generic components. (See *PJ* 6.5 for a discussion of Ada generics.)

In addition, Ada implements the reliability goal through strong data typing, built-in exception handling, strict scope and visibility rules, and formal program structures. Ada also provides a capability for data and structural abstraction and component reusability. The language was designed to be used in "mission critical" embedded RTSes such as space vehicles, radar guidance instruments, and your 1993 model pacemaker. By "mission critical" we also mean, reliability critical. However, any software system can benefit from an experience of the rigorous demands on embedded RTS software.

We will not be able to explore all of Ada's reliability features in this one short article. Consequently, we shall focus on two that support the reliability goal: functional types and exception handling. In future installments we'll examine other Ada features.

Functional Typing

Many programmers still begin the debugging process by searching for errors in program logic. Only after prowling through the code at length, do they discover some problem with the data definition (or the data), not the logic. Every programmer knows that careful attention to data structures is fundamental to good software development, so fundamental that it's often overlooked.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

On a recent sortie to the computer section at Printer's Inc., bookstore in Palo Alto, I overheard a young man say to his companion, "Ada, just say no." As an Ada evangelist, I could not resist challenging him. He explained that he'd just finished a year and a half Ada project and came away hating Ada. I drew him out and discovered that neither he nor the others on the project had really taken the time to learn Ada. He said, "I guess we just fought strong typing, instead of accepting it." Moreover, his project team hadn't really learned to create and use generics until the project was almost finished. This seems to be typical of old-timers new to Ada. For some reason, Ada is more hospitable to virgin programmers. Some experienced programmers are "code macho"; they seek to prove themselves to be among the software elite by producing code at the most primitive level possible.

In earlier programming languages such as FORTRAN and COBOL, and in low-level languages such as Assembler, data structures are declared according to their physical representations in memory. More recent languages such as Pascal, Ada, and Modula-2 use logical and functional data representation. "Physical" versus "logical" can be abstracted to "form versus function." A logical (type) data declaration can ignore the bit structure of a data item as long as we know it corresponds to our program requirements. For example, in this declaration:

```
EIN, ZWEI, DREI : INTEGER;
```

it is necessary to know only the upper and lower ranges of INTEGER, not how an integer is represented in memory.

Reliance on physical definition of data often leads to trouble: too much flexibility for assignment and control structures. An assignment statement may be coded independently of the class of data on which it operates. We can, in effect, perform reasonable operations on illogical data items.

In a theoretical assembler language we may add apples to oranges:

```
MOV DL,03  ; load apple  count (3)  in DL  register.
MOV AL,06  ; load orange count (6)  in AL  register.
ADD AL,DL  ; add  apple register to orange register.
```

Type Classifications

physical type	=	define data on the basis of its physical representation on the object computer
logical type	=	the general form of the data: definitions such as INTEGER, FLOAT, or STRING are logical types; the next level of abstraction above physical type
functional type	=	define the data based on how it's used in a program; a level of abstraction higher than logical type
encapsulated type	=	a type defined in a separate package and accessed via the Ada **with** clause
abstract type	=	an encapsulated type; operations on this type are defined within the scope of a package; no operations are permitted except those defined in the package

This may also occur in languages such as FORTRAN or COBOL. Even C and Forth permit too much access to the physical representation of data. In those languages we must create validity checking routines in our program logic. In a language such as Pascal, Modula-2, or Ada, language inherent features prohibit us from mixing data types. The C language does have a **typedef** feature, but it is not enforced by the language, and many programmers ignore it. Moreover, some compilers ignore it. Forth does permit the programmer to build high-level abstractions, but only after creating a low-level foundation.

Ada provides us with methods for creating functional types and subtypes from logical and physical types. In the example above we might do the following:

```
declare
   type APPLE_TYPE  is new INTEGER;  -- derived type
   type ORANGE_TYPE is new INTEGER;  -- derived type
   GREEN_APPLES : APPLE_TYPE;        -- define object type
   RED_APPLES   : APPLE_TYPE;
   ORANGES      : ORANGE_TYPE;       -- define object type
   APPLE_TOTAL  : APPLE_TYPE;        -- define object type
   ORANGE_TOTAL : ORANGE_TYPE;       -- define object type
begin
   APPLE_TOTAL  := RED_APPLES + ORANGES;       -- illegal
   ORANGE_TOTAL := ORANGES + GREEN_APPLES;     -- also illegal
   APPLE_TOTAL  := RED_APPLES + GREEN_APPLES; -- OK
end;
```

In this trivial example, we have created functional rather than physical representations of the data. This illustrates a central idea of strongly typed languages. The functional type is an abstraction of the physical type. Moreover, the abstraction enforces a consistency of operations and values for that type.

Ada has available many predefined logical types. In addition, an Ada programmer can create new functional types, derive functional types from other types, hide the implementation of a functional type, and create a complete new set of restricted operations on a new type.

It is appropriate for us to know that Ada does have a mechanism for gaining access to physical type representations when necessary. Chapter 13 of the *Ada Language Reference Manual* (*LRM*) defines an orderly way to transform low-level requirements into high-level representations. The chart at the top of the page illustrates the hierarchy of predefined Ada data types.

Pascal programmers will notice something familiar in Ada data types. This is because Ada is derived from Pascal (which is in turn derived from ALGOL). We can indulge ourselves in one small example of how Ada extends a Pascal concept. In Pascal we may create something called a "subrange" by defining a new type:

```
Const
      MAX_APPLES  = 100;
      MAMAX_ORANGES = 100;
Type
      MAAPPLE_TYPE  = 1..MAX_APPLES;
      MAORANGE_TYPE = 1..MAX_ORANGES;
```

In this case, Pascal allows us to create these subranges of the predefined type INTEGER. Ada permits a similar construct called a "subtype." The subtype is more explicit and permits a deeper hierarchy of decomposition. First we define a new type:

```
MAX_APPLES  :constant := 100;
MAX_ORANGES :constant := 100;
type APPLE_TYPE  is range 1..MAX_APPLES;
type ORANGE_TYPE is range 1..MAX_ORANGES;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Ada Leadership

New products

The Booch Components

The Booch Components is a carefully engineered library of over 500 highly reusable components. Standard abstract data types are supported, such as stacks, queues, trees, graphs, and lists. Several utility packages provide sorting, searching, pattern matching and other capabilities. The components are bundled with the best-selling textbook *Software Components With Ada* by Grady Booch. Meridian is the exclusive distributor of The Booch Components for Apple Macintosh, IBM PC/compatibles, and Sun Microsystems workstations.

Application Interface Management System (AIMS)

AIMS is a productivity-enhancing set of tools and libraries for constructing application user interfaces with Meridian AdaVantage for MS-DOS. AIMS enables programmers to quickly define configurable interactive applications such as point-of-sale registry, inventory control, reservation management, and scientific data entry. AIMS assists developers by automating routine coding to do formatting, justification, editing, and data validation.

The Meridian Ada technology and market leadership tradition continues with the introduction of The Booch Components™ and the Application Interface Management System.

At Meridian, we develop and market products with our customer and our customer's future in mind. We believe today's state-of-the-art productivity solution must integrate with tomorrow's. Meridian is committed to an evolving, modular product architecture, incorporating tools and libraries to assist the Ada developer.

Meridian's Ada products are available for IBM PC/compatibles and Apple Macintosh. In early 1989 we will introduce product lines for Sun Microsystems workstations and DEC VAX hosts.

product profile

Professional Development Kit — A cost-effective Ada Programming Support Environment for IBM PC/compatibles. Includes AdaVantage Optimizing Compiler, Debugger, Ada Developer Interface, and DOS and Utility Libraries.

AdaVantage v2.2 Optimizing Ada Compiler — Optional support for protected mode is now available with AdaVantage for IBM PC/compatibles. Allows access to the full address space of 80286.

AdaVantage for the Apple Macintosh is the only Ada compiler available for the native Macintosh Operating System.

AdaVantage Debugger — The AdaVantage Debugger is a complete, Ada source-level debugger.

Ada Developer Interface — APSE interface with "point and click" interaction. Includes Ada editor, formatter, "make" function, and "what-if" analysis.

Run-Time Customization Library — Allows preparation of Ada application programs for execution on an 80X86-based embedded system.

Meridian SHARP — On-line Ada reference manual and Ada software engineering courseware for the Mac.

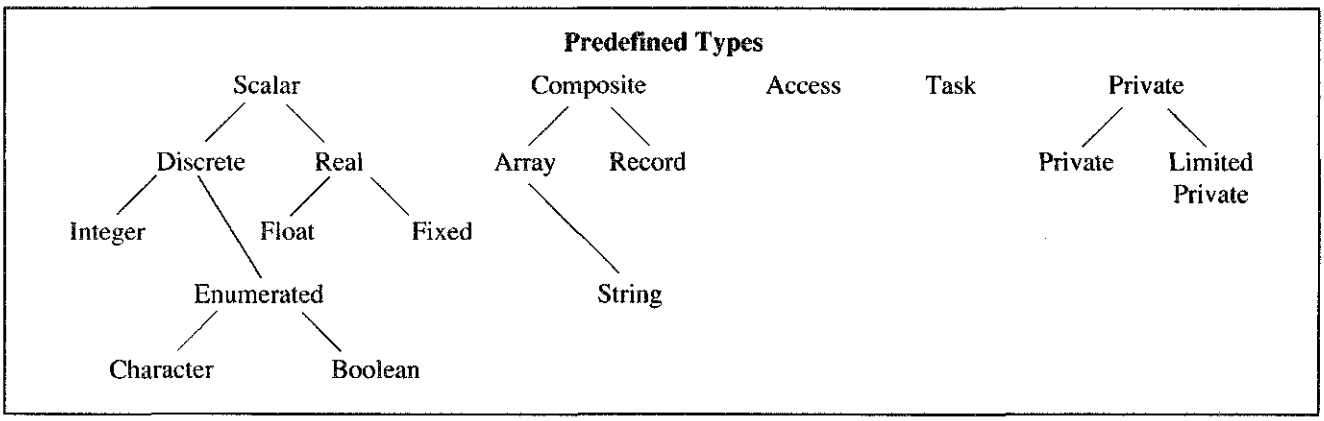
MERIDIAN
SOFTWARE SYSTEMS, INC.

23141 VERDUGO DRIVE • SUITE 105 • LAGUNA HILLS, CALIFORNIA 92653
800/221-2522 (Outside California) • 714/380-9800 (Inside California)
Telex: 650-268-0547 MCI • Fax: 714/380-1683

☐ The information contained herein is subject to change without notice. ☐ AdaVantage, AdaTraining, AdaDesigner and AdaStarter are trademarks of Meridian Software Systems, Inc. References to other computer systems use trademarks owned by the respective manufacturers. ☐ Copyright® 1988 Meridian Software Systems, Inc. All rights reserved.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Number 235 on the Reader Service Card



then we define subtypes:

```
subtype FIRST_FIVE_APPLES  is APPLE_TYPE  range 1..5;
subtype FIRST_FIVE_ORANGES is ORANGE_TYPE range 1..5;
```

We can declare a subtype in terms of a previous subtype:

```
subtype FIRST_APPLE  is FIRST_FIVE_APPLES  range 1..1;
subtype FIRST_ORANGE is FIRST_FIVE_ORANGES range 1..1;
```

This is a strange example, but it illustrates how we can create a hierarchy of subtypes. When subtypes are used in calculations, constraint errors will be raised when data is out of range. But related subtypes may be mixed with parent types in Ada expressions.

Aficionados of object-oriented design will notice a kind of reverse "inheritance" scheme. Here, the parent type inherits the attributes of the child subtype. The subtype inherits all the operations permitted on the parent type.

We won't try to examine every data type available in Ada in this short article. Our focus is on Ada's charter to make it easier to create reliable software. In this regard, Ada provides us with a unique feature: private types. Private types allow us to raise the level of abstraction even beyond functional types.

Private Types

There are two kinds of private types: *private* and *limited private*. Any type may be encapsulated in an Ada package. When we encapsulate a private type in a package, we substantially raise the level of abstraction:

```
package FRACTION_MATH is
   type FRACTION is private
          -- more type definitions
   function  SUM(X1,X2 : FRACTION) return FRACTION;
   procedure COMPOSE(N,D :INTEGER; F : out FRACTION);
          -- more functions and procedures
private    -- the private part of the package specification
   type FRACTION is
      record
          NUMERATOR   : INTEGER;
          DENOMINATOR : INTEGER;
      end FRACTION;
end FRACTION_MATH;
package body FRACTION_MATH is
          -- algorithms for functions and
          -- procedures are found here
end FRACTION_MATH;
```

Here we have created a new "abstract" type named FRACTION. The only operations allowed on FRACTION outside the package body of FRACTION_MATH are: test for equality or non-equality (no > or < permitted), assignment, declaration of objects of type FRACTION, and passing as parameters to procedures in the package name FRACTION_MATH. This is an implementation of the principle of "information hiding" discussed in our first article on Ada.

In this case, the components of FRACTION (NUMERATOR and DENOMINATOR) are not directly accessible by any other program. We could refer to them only at the higher level of abstraction, named FRACTION. In our procedure to use package FRACTION_MATH we might say,

```
with FRACTION_MATH; use FRACTION_MATH;
procedure MY_MATH is
  N1, D1, N2, D2 : INTEGER;
  ANSWER,Y1, Y2 : FRACTION;
begin
  COMPOSE(N1, D1, Y1);
  COMPOSE(N2, D2, Y2);
  ANSWER := SUM(Y1,Y2);
end MY_MATH;
```

which would perform the operation named SUM on Y1 and Y2 and put the result in ANSWER. Only the procedures of package FRACTION_MATH have access to the components of Y1, Y2, and ANSWER. Therefore, we have a procedure in FRACTION_MATH named COMPOSE, which creates two data items of type FRACTION from INTEGERs.

The other private type, limited private, is an even higher level of abstraction. It does not permit assignment or comparison. A parameter may be passed, but all operations are restricted to those defined in the package in which the limited private item is defined. For example:

```
package RESOURCE_MANAGER is
   type RESOURCE is limited private;
   procedure ALLOCATE  (R : out RESOURCE);
   procedure ACTIVATE  (R : in  RESOURCE);
   procedure DEALLOCATE(R : in  RESOURCE);
private
   type RESOURCE is new INTEGER range 1..10;
end RESOURCE_MANAGER;
package body RESOURCE_MANAGER is
   procedure ALLOCATE  (R: out RESOURCE) is
     -- allocate one of the ten resource indicators
     -- in the private part of the package specification
   end ALLOCATE;
   procedure ACTIVATE  (R: in  RESOURCE) is
     -- invoke the use of one of the resources
   end ALLOCATE;
   procedure DEALLOCATE (R: in RESOURCE) is
     -- return the resource to a useable state
   end DEALLOCATE;
```

To use RESOURCE_MANAGER we might say:

```
with TEXT_IO; with RESOURCE_MANAGER;
procedure HROTHGAR is
   SEMAPHORE : CHARACTER;
   DUNGEON   : RESOURCE_MANAGER.RESOURCE;
   DRAGON    : RESOURCE_MANAGER.RESOURCE
begin
   loop
       TEXT_IO.GET(SEMAPHORE);  -- character input
       case SEMAPHORE is
           when "N" =>
                 RESOURCE_MANAGER.ALLOCATE(DUNGEON);
                 RESOURCE_MANAGER.ACTIVATE(DUNGEON);
                 RESOURCE_MANAGER.DEALLOCATE(DUNGEON);
           when "Y" =>
                 RESOURCE_MANAGER.ALLOCATE(DRAGON);
                 RESOURCE_MANAGER.ACTIVATE(DRAGON);
                 RESOURCE_MANAGER.DEALLOCATE(DRAGON);
           when others
               -- statements to deal with other conditions
       end case;
   end loop
end HROTHGAR;
```

In this example, the functional type, RESOURCE, may be used in HROTHGAR only as a parameter to one of the procedures in RESOURCE_MANAGER. In other words, no operations are allowed on data of type RESOURCE outside the package, RESOURCE_MANAGER. Therefore, an operation such as

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
DUNGEON := DRAGON;
```

is illegal even though both objects are of the same type.

Designing with private and limited private types can be most effective in creating more reliable software. Not only do we implement the principle of information hiding with private and limited private types, but we also enhance the cohesiveness of the executable component in which that type is encapsulated and raise the level of abstraction.

Additional Types

We shall briefly discuss three more types that are typical of Ada: enumeration, access, and arrays. The *enumeration* type gives us a method of naming the elements of a group. In Ada we can process the elements of a group as if they were an array. Here are some examples of the enumeration type:

```
type DAY_TYPE is (SUN, MON, TUES, WED, THURS, FRI, SAT);
subtype WORK_DAY_TYPE is DAY_TYPE range MON..FRI;
type COUNT is (ICHI, NI, SAN, YON, GO, ROKU, SHICI, HACHI, KU,
  JU);
```

We could have a logical construct such as:

```
declare
  for JAPANESE_NUMBER in COUNT
      loop
          NEW_LINE;
          PUT(JAPANESE_NUMBER);
      end loop;
end;
```

The use of enumeration types, especially when combined with limited private or private capability, provides a powerful method of creating new functional data types.

For programmers accustomed to pointers in C or Pascal, *access* types serve the same purpose:

```
type LIST-POINTER is access INTEGER;
```

We'll devote an entire article to the access types in a future issue. In fact, this topic is so important that it deserves to be examined independently of Ada so that we can see its applications in many different languages.

The *array* is basic to many well-constructed algorithms. In principle, almost any data structure can be reduced to some sort of array. Ada provides a capability for both constrained and unconstrained arrays:

```
type ARRAY_TYPE is array (INTEGER range 1..5) of INTEGER;
type U_ARRAY_TYPE is array (INTEGER range <>) of INTEGER;
```

The <> is read "box" and designates an unconstrained array. We must constrain it in a later declaration. The unconstrained array is not found in standard Pascal implementations. In fact, Dr. Wirth, creator of Pascal and Modula-2, reportedly disapproves of unconstrained arrays. There is one predefined use of the unconstrained array:

```
type STRING is array (POSITIVE range <>) of CHARACTER;
```

which gives us a predefined data type of STRING.

There are many more variations on data typing in Ada. We'll see other examples in our future explorations of Ada.

Exception Handling

A parting salutation of one of the Native American tribes was "Stay well and don't go crazy." This would be an appropriate final directive to a newly inaugurated software product.

Since Ada is designed for implementation of mission critical real-time systems, it is not enough to have strong typing unless we also have an inherent corrective maintenance capability.

Experienced programmers know that exception processing is one of the most important components of their produc-

DISK FORMAT CONVERSION

Xeno Copy-PC

$79.95 + $5.00 S/H
Sales Tax if CA.

PC-DOS program lets your PC

VISA MasterCard UPS COD

READ / WRITE /
FORMAT / DUPLICATE
Disks from over 300 other micros

Upgrades available from all previous versions
for only $25.00 Call for Authorization

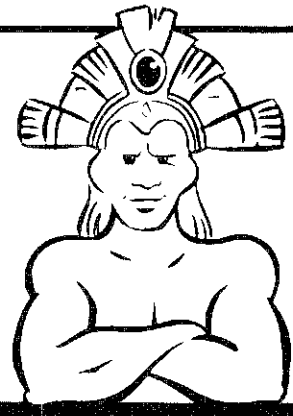
To Order Contact:

XENOSOFT™

2210 Sixth Street, Berkeley, CA 94710
(415) 644-9366

Number 123 on the Reader Service Card

EVERLOCK



COPY PROTECTION

Designed for user-transparency, clone compatibility & strength. It features:

- no need for damaged media or I/O plugs
- supports all Hard & Floppy disk formats
- file-server networks supported
- variable number of installs (0-99)
- create demos with remote unlock option
- allows protected upgrades by modem BBS
- limit program use by date or count

$195 Starter Kit or $495 with NO meter counts. Free info & demo disk available

AZ

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085

VISA MasterCard

(800) 227-0644 (816) 776-2700

Number 51 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tion code. It is one of Ada's more significant contributions to reliability. Unlike other strongly typed languages, Ada includes a well-developed exception handling feature. When a run-time error occurs due to a type mismatch or constraint error, an Ada programmer can provide for it.

Let's imagine that we have a procedure P1 that includes an exception handling routine. The exceptions are E1, E2, and E3. The exception handling procedures are EP1, EP2, EP3, and EP4:

```
procedure P1 is
    -- some specification goes here
begin
    -- some procedural code goes here
exception
    when  E1     => EP1;
    when  E2     => EP2;
    when  E3     => EP3;
    when others => EP4;
end p1;
```

Notice that we have a nice little catch-all capability, **others**, to make sure we have covered all the possibilities.

Ada has a lot of built-in exceptions, and many Ada packages contain their own exception declarations. The language definition predefines packages: STANDARD, SYSTEM, IO_EXCEPTIONS, TEXT_IO, DIRECT_IO, SEQUENTIAL_IO, CALENDAR, and a few others. In each of these packages there are predefined exception items. For example, package STANDARD includes

```
CONSTRAINT_ERROR, NUMERIC_ERROR, PROGRAM_ERROR,
STORAGE_ERROR, and TASKING_ERROR.
```

and package TEXT_IO includes

```
STATUS_ERROR, MODE_ERROR, NAME_ERROR, USE_ERROR,
DEVICE_ERROR, END_ERROR, DATA_ERROR, and LAYOUT_ERROR.
```

The programmer can, and should, include exception handling routines as a matter of standard practice. The problem, of course, is what should those routines do? The solution is left as an exercise for the student.

One nice feature of Ada is the ability to declare our own exception items. These are declared in much the same manner as type declarations but include the word **exception**. Using the example above we could have declared

```
E1, E2, E3  : exception;
```

then in our code,

```
if <some condition> then raise E1;
```

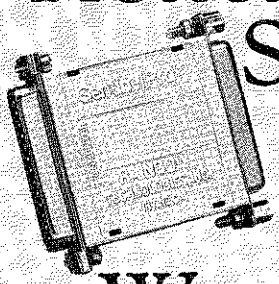
This would have caused processing to transfer to the exception handling "frame," and then transfer to the appropriate procedures for an E1 exception.

Sometimes we want to have the exception handled and continue with the standard processing cycle. Using the example above, we could have

```
procedure P1 is
   E1, E2, E3 : exception;       -- declare three exceptions
   -- additional declarations
begin
   loop        -- unconditional loop starts here
        begin -- start an Ada block
              -- some code
              -- exception testing and raising
              exit; -- exits loop beyond exception handling
        exception
              -- exception handling statements
        end;  -- end of the block within the loop
   end loop;  -- end of the loop
   -- more P1 code
   -- more P1 code
end P1;
```

We included an **exit** statement to make the code more nimble. This allows us to create a loop that includes our exception handler and to skip over the exception handler when everything is all right. Moreover, we structured our exception han-

Suppose You Protected Your Software & The Silence Was Golden.



Software protected by Rainbow Technologies' Software Sentinel™ has a sound of its own: Golden. And unlike copy-protected software, very silent.

Golden because your development investment is protected. Down to each and every copy you sell. To give you the long-term solution to revenue loss. And the sure way out of the worldwide free-distribution network.

Silent because Software Sentinel protection is transparent. With easy-to-live-with reliability. And no impact on the user or your software's performance. Which means all you hear are the nods of approval.

The Software Sentinel key is shipped with the software and easily installed on the PC's printer port. Protected software will run only when it's installed. No key – No use. It's that simple.

Key Sentinel Family Features.

▫ "Execution control" technique prohibits unauthorized use of software ▫ Permits unlimited backup copies ▫ Virtually unbreakable ▫ For DOS, OS/2 and XENIX ▫ Pocketsize – Transportable ▫ Transparent operation ▫ Use with a hard disk

SentinelPro™

▫ Runs under DOS, OS/2 and Xenix, on IBM PC/XT/AT, PS/2 and compatibles
▫ Algorithm technique (Nanny a fixed response)
▫ Parallel and serial port versions
▫ Minimal implementation effort
▫ Hooks for C language

Sentinel-C

▫ Protects multiple packages with one device
▫ 126 bytes of non-volatile memory programmed before shipment of the software
▫ Rainbow supplies the Lockbox 2 utilities adapter for programming the unit
▫ Flexible configuration interfaces included
▫ Runs under DOS, OS/2 and Xenix on IBM PC/XT/AT, PS/2 and compatibles
▫ Parallel port only
▫ Development kits available

RAINBOW TECHNOLOGIES

Number 222 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

dling routines so that we could execute them and return to the main processing routine without having to abort the program. This example is worth studying because it contains some useful features of the language.

Ada will give priority to monitoring its predefined exceptions such as

```
CONSTRAINT_ERROR, NUMERIC_ERROR, PROGRAM_ERROR,
STORAGE_ERROR, or TASKING_ERROR
```

The runtime system will test for these errors whether the programmer includes them in his exception handler or not. If there is no specification of what to do for an exception, the program will ABEND (abnormal end of job). Ada implementations include exception packages, notably IO_EXCEPTIONS, that can be used in your programs. Again, the programmer must decide which exception handling conditions to include in the program.

Fortunately, it is not necessary to specifically indicate every exception. The following syntax is acceptable:

```
begin
        -- some program statements
        -- or exception raising code
exception
        when DATA_ERROR =>
             PUT("Invalid entry, try again");
        when others
             PUT("Good Grief! That was really stupid!");
             PUT("Try Again -- and be more careful.");
end;
```

The use of **others** permits us to catch all of the exception conditions we don't want to specify.

The availability of built-in exception handling enhances our ability to program for reliability—at least from the point of view of fault tolerance.

In future articles on Ada we'll examine more of the language environment's emphasis on reliability. This will be especially important when we look at how Ada is used for real-time systems (RTS) environments. Meanwhile, try to acquire one of the low-cost Ada compilers for your PC or Macintosh and experiment with converting some of your old Pascal programs to Ada. ♦

Richard D. Riehle is an instructor in Ada at Merritt College in Oakland, and Foothill College in Los Altos, California. He is also an independent computer consultant. Mr. Riehle's training organization ICE (for Intensive Computer Education) conducts seminars on a variety of computer subjects. Mr. Riehle has over twenty years in software development, including design, programming, and project management. He is currently working on a book, From COBOL to ADA, *a tutorial for experienced business programming professionals. Readers may contact Mr. Riehle at 1090-105 Tanland Drive, Palo Alto, CA 94303, (415) 858-1267.*

Did I Really Want to Hear This?

The luncheon speaker at a user's conference I attended in the mid 1960s represented a commercial aircraft manufacturer. During the Q&A session following his talk, he was asked what he thought of the current state of computing technology. His reply? "If we built our airplanes to the same level of reliability at which computers are built, one of you at each table in this room would not get home from this conference."

The tables each seated eight.—*W.G. Madison* ♦

ADD TO THE POWER OF YOUR PROGRAMS WHILE YOU SAVE TIME AND MONEY!

CBTREE does it all! Your best value in a B+tree source!

Save programming time and effort.
You can develop exciting file access programs quickly and easily because CBTREE provides a simple but powerful program interface to all B+tree operations. Every aspect of CBTREE is covered thoroughly in the 80 page Users Manual with complete examples. Sample programs are provided on disk.

Gain flexibility in designing your applications.
CBTREE lets you use multiple keys, variable key lengths, concatenated keys, and any data record size and record length. You can customize the B+tree parameters using utilities provided.

Your programs will be using the most efficient searching techniques.
CBTREE provides the fastest keyed file access performance, with multiple indexes in a single file and crash recovery utilities. CBTREE is a full function implementation of the industry standard B+tree access method and is proven in applications since 1984.

Access any record or group of records by:
- Get first
- Get previous
- Get less than
- Get greater than
- Get sequential block
- Get all partial matches
- Insert key and record
- Delete key and record
- Change record location
- Get last
- Get next
- Get less than or equal
- Get greater than or equal
- Get partial key match
- Get all keys and locations
- Insert key
- Delete key

Increase your implementation productivity.
CBTREE is over 8,000 lines of tightly written, commented C source code. The driver module is only 20K and links into your programs.

Port your applications to other machine environments.
The C source code that you receive can be compiled on all popular C compilers for the IBM PC and also under Unix, Xenix, and AmigaDos! No royalties on your applications that use CBTREE. CBTREE supports multi-user and network applications.

CBTREE IS TROUBLE-FREE, BUT IF YOU NEED HELP WE PROVIDE FREE PHONE SUPPORT.
ONE CALL GETS YOU THE ANSWER TO ANY QUESTION!
CBTREE compares favorably with other software selling at 2,3 and 4 times our price.
Sold on unconditional money-back guarantee.
YOU PAY ONLY $159 - A MONEY-SAVING PRICE!
TO ORDER OR FOR ADDITIONAL INFORMATION
CALL 1-800-346-8038 or (703) 847-1743
OR WRITE

PEACOCK SYSTEMS, INC.

NOW! Variable length records.

Peacock Systems, Inc., 2108-C Gallows Road, Vienna, VA 22180

Number **216** on the Reader Service Card

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Developer's Data Manager

AXOS™

It Speaks Your Language!

BASIC • COBOL • C • FORTRAN • PASCAL • ASSEMBLER

A New Standard in Access Method Technology

AXOS is a *fast* full-featured, key-indexed file management system designed to provide application programmers and software developers a flexible and versatile data storage and retrieval mechanism with the least amount of overhead. It can be used in either single or multi-user LAN environments on the IBM PC PS/2 and compatibles. As AXOS can be called from any programming language through a common communication protocol, it allows one to manipulate data in a manner not possible with other DBMSs or access methods. AXOS also offers a full complement of file mangement functions, system management functions, productivity tools and debugging aids designed to ensure easy and accurate development and support of microcomputer applications systems.

FEATURES

- **SECURITY:** LAN computer name and password protection. Eight types of file I/O security by computer.
- **AUDIT:** Complete (update and retrieval) or partial (retrieval) capture of all AXOS activity, available with selection criteria both online or hardcopy.
- **DATA INTEGRITY:** Ensures full data integrity in the event of a program, system or power failure.
- **INTERACTIVE MONITOR:** Instantaneous interactive monitoring of AXOS activity as a program is executing.
- **FILE CONVERSION:** Convert or merge existing files of other types into any one of the AXOS file formats available.
- **FILE LIST:** Produces a formatted report of the contents of all AXOS file types and a variety of DOS sequential file types.
- **PROGRAM INVOKED ERROR ROUTINE:** Used by application programs for processing unexpected I/O return codes.
- **RECORD ENCRYPTION AND COMPRESSION:** All records contained on AXOS files are encrypted and compressed so as to reduce disk storage requirements.
- **PAGE BUFFERING:** AXOS builds and manages all of its own buffer space so as it minimize disk I/O and decrease processing time in both single user on multi-user (LAN) environments.
- **PARAMETERS:** All of the AXOS keyboard commands utilize parameters entered at the time that the command is executed. This provides a level of flexibility wherein a command may be tailored to meet specific user requirements.

SPECIFICATIONS

- **Languages Supported:**

  BASIC COMPILED BASIC COBOL
  FORTRAN C ASSEMBLER PASCAL

- **Local Area Networks Supported:**

  IBM® PC NET AT&T® StarLAN
  3Com® 3+Share Novell® Netware

- **File Types Supported .....:** Sequential, Single Index, Multi Index (cluster)
- **Maximum Indexes .....:** 64 per file
- **Maximum Key Length .....:** 240 bytes for unique keys, 236 for non-unique keys
- **Maximum Record Size .....:** 64,000 bytes
- **Maximum Records Per File .....:** 16,777,216
- **File Capacity .....:** 4.2 billion bytes
- **Maximum Files & Indexes allowed open at one time .....:** 253
- **Operating System Requirement :** DOS 2.1 or above, 128 - 256KB of memory

Come To Grips With The Power Of AXOS

"McSoft's AXOS is not for the faint of heart - it clearly is aimed at the professional developer who is building end-user applications or intermediate development systems." PC Tech Journal 10/88.

McSoft INCORPORATED™

AMEX/COD/MC/VISA

$595 Includes 30 days free technical support
Extended technical support available via HOTLINE

And there are no royalties on your AXOS applications!

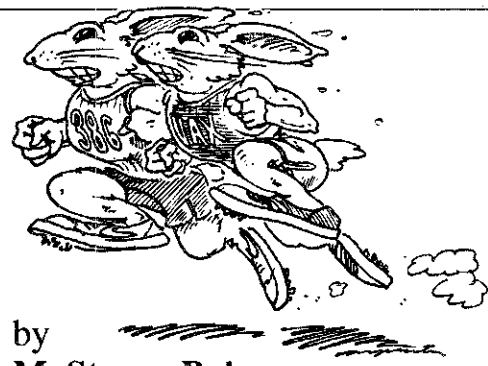
FOR MORE INFORMATION OR TO ORDER CALL (404) 237-4169

P.O. Box 12044 • Atlanta, GA 30355-2044

Number **230** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Crunching Numbers At Blazing Speed



by
M. Steven Baker

Although PC software has supplanted many applications from the minicomputer and mainframe world (word processing, spreadsheets, databases, etc.), large, numerics intensive applications have remained in the realm of the big machine. It's only recently that they've begun to migrate to workstations. If a simulation took 90 minutes of CPU time on a VAX, it would take forever on a PC. The advent of fast 32-bit microprocessors (80386 and 68020) alone was not sufficient to make the PC competitive with larger machines. The 32-bit microprocessors provided the ability to address large blocks of memory, but most mainframes and minicomputers have floating point accelerators designed to speed real math applications. Microcomputers still needed fast floating point coprocessors if they were to move numerics intensive applications off the mainframe. To be useful, the new hardware had to be well supported by PC software.

It's starting to come together. In this article I'll explore the floating point hardware and software now available for the PC. I'll describe my experiences using the 80387 and the Weitek 1167 and 3167 math chips on 80386 machines and two coprocessor boards for the PC: (1) A.I.Architects' Hummingboard—80386/80387; and (2) Definicon's DSI-780 boards—68020/68882. I'll include brief reviews of C and FORTRAN compilers and software development tools that support large numerics applications on the PC.

Take Me Back in Time: Intel Chips

Although the 8087 was the first microcomputer math coprocessor to achieve widespread success, Intel and other manufacturers did make earlier math chips. Those of you whose memories go back a decade will remember that the Compupro S-100 boards supported two different early math chips (Intel 8232 and 8231A) for the 8085 and Z-80. Those early math chips were generally I/O port mapped, and not directly supported by the CPU (you had to load instructions and data in 8-bit chunks by reading and writing to a port). The 8087 math coprocessor, though, was designed for the 8086 architecture and is directly supported by the CPU and the 8086 instruction set.

The 8087 has a stack-based architecture with 8 80-bit wide registers (10-byte) providing extended precision. A number of data formats are supported by the 8087 although most PC software uses the 8087 registers for single (4-byte) and double precision (8-byte) floating point calculations implementing the IEEE floating point standards (which have continued to evolve). The 8087 communicates with the 8086/8088 synchronously, so the 8087 must operate at the same clock speed as the 8086. Because of the direct connection, no overhead is involved in 8086/8087 communications.

The basic design and instruction set of the 8087 has passed on to the 80287 and 80387 math chips for downward compatibility with the 80x86 architecture (Hal Hardenbergh's "code museum" phenomenon). One major difference is that the 80287 and 80387 chips operate asynchronously from the CPU, so the math chip may operate at either a faster or slower speed than the CPU. This asynchronous connection was originally a product of two considerations: (1) The 80286 and 80386 chips have memory protection, and if the 80287 chip were to be directly connected, the equivalent memory protection logic would have to be duplicated on the math chips as well; and (2) 80286 CPU chips were first available at 6 through 10 MHz clock speeds, but the original 80287 math chips would not operate at these speeds. Hence, IBM ATs and most AT clones clock the math chip at two-thirds the speed of the 80286. For this reason, a turbo XT (fast 8086/88 or NEC V30 with an 8087) such as MicroWay's Number Smasher board will usually speed by an unmodified AT in math intensive computations. Also, be warned that compared to the original 8086/8087, this asynchronous communication of the 80287 and 80387 does have overhead associated with it.

The 80386 chip was designed to support any of three math chips: an Intel 80287 or 80387 coprocessor or a Weitek 1167 math chip set. The decision to support the 80287 was a stopgap since the 386 chip was available considerably before any working 387 chips. The 80287 chip used with an 80386 now makes for a serious mismatch. While the 80386 chip is running at 16 or 20 MHz, the 80287 chip is generally clocked at the AT bus speed (8 or 10 MHz), half the speed of the CPU. Again, a fast turbo XT might outperform an 80386/80287 for some numerics intensive applications in the real mode. The 8087 and 80287 only partially supported some transcendentals (sine, cosine, etc.), so transcendentals needed to be supported with software subroutine calls. The 80387 added better support for these instructions, allowing faster inline code to be generated. Intel has been a strong supporter of IEEE 754 standards development, so the 80387 dropped support for some minor features of the 80287 in order to correspond with the IEEE standard. The 80287 and 80387 generally execute the same instructions in the same number of clock cycles except that the 80387 is clocked at 16–20 MHz versus the 6—12 MHz of the 80287. So in real time, the 80387 is typically about twice as fast.

The Next Age: Weitek Coprocessors

One weakness of the 80386 is that the speed of the Intel math coprocessors (originally the 80287) were not competitive with the floating point accelerators available for minicomputers and worksta-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tions. Weitek, a Silicon Valley manufacturer of high end math, vector, and graphics processors, worked with Intel to design an 80386 interface to their math chip set used in various workstations. Intel and Weitek jointly financed the Weitek interface development. The Weitek WTL 1167 is a high-performance floating point coprocessor for the 80386. Currently, many 80386 machines support both the 80387 and Weitek math chips via the Extended Math Coprocessor (EMC) socket, a 121 pin superset of the 80387 chip. The original Weitek chip set is called the 1167 and consists of three chips—the 1164 multiplier, the 1165 ALU, and the 1163 80386 interface chip that also contains the register file. The first two chips form the core of the floating point accelerators used in Sun, Apollo, and other workstations.

The 1167 chip set comes on a small board that plugs into the 121 pin EMC socket, usually with a socket for the 80387. The 1167 is available directly from computer manufacturers like Compaq and also from MicroWay as their mW1167 boards. MicroWay has been supplying math coprocessors and supporting software for the PC for many years. They manufacture and sell Weitek 1167 boards for most 80386 computers including Compaq, Everex, Acer, Tandy, AT&T, and other 80386 machines rated at 16 and 20 MHz speeds. This summer, Weitek announced the fall availability of their "Abacus" chip—the 3167, a single chip 121 pin implementation of the 1167 with some internal speed improvements. The 3167 will be available in 20 and 25 MHz versions, usually on a very small daughterboard to allow plugging in an 80387. The Compaq 386/25 actually has two separate sockets—one for the 3167 and one for the 80387. Weitek and Intel do not consider the 1167 and 3167 to be in competition with the 80387; the 1167 is targeted to provide faster numerics performance for high-end applications.

The Weitek 1167 and 3167 were designed more as vector processors than as general math coprocessors. The WTL 1167 provides the four basic math functions plus negation, absolute value, comparison/testing, data movement, and format conversion functions. The WTL 1167 consists of 32 single precision floating point registers (ws0–ws31) that can also be combined and used as 16 double precision registers, along with control and status registers. The chip set supports fast execution times for addition, subtraction, multiplication, and division. Single precision math functions take substantially less time than double precision. Transcendental functions are handled by a software subroutine library provided by Weitek at a nominal cost. The WTL 1167 (and 3167) is memory mapped into the 80386 address space—it masquerades as a 64K block of memory in the 80386's physical memory space. On the PC its physical address is usually 0C0000000h; however, supporting software usually maps these addresses elsewhere. There are currently two conventions for the mapping of the Weitek on the PC in the protected mode: (1) The Phar Lap convention maps the Weitek to FS:0 (that's segment register FS), and (2) the Unix convention maps the Weitek starting at linear address 0FFC00000h. If the paging capabilities of the 80386 are used, the Weitek chip can also be mapped to that mythic 64K block that begins at the just below 1 megabyte address in the virtual 8086 mode for standard DOS programs (real mode)—see Jake Richter's "Another 64K of Memory for DOS Applications" in this issue.

The Weitek registers are also mapped in this same 64K memory block. Therefore, context save and restore can use the fast 80386 REP MOVSD (move double word) instruction to copy or replace a block of register values. Instructions are executed by MOV instructions to this memory block. Weitek provides a set of macros for instruction mnenomics. The instructions are mapped for speed in vector operations so that multiplying an array in memory with an array in the Weitek registers can also make use of the same fast MOVSD instruction—except that now we point to a memory location for the Weitek multiply instruction. The lower 16 bits of the address encode the WTL instruction type and both the source and destination registers. For example, the WS_MUL instruction would start at physical address 0c0000800h, with an offset for the Weitek registers to be manipulated. The following sample instruction sequence multiplies each element of a floating point array VECTOR

Intel 80386 ERRATUM #21:

Coprocessor Malfunctions with Paging Enabled

B step 80386 chips have a bug that can appear only when executing 80387 instructions and paging is enabled. The problem can occur under certain conditions of memory wait states and HOLD requests. If paging is enabled and the 386 prefetch unit requests a memory read cycle at the same time that the 80387 requests an operand transfer, the 386 will drive out an incorrect address leaving the 387 in an indeterminate state. Basically, each chip is waiting for the other to complete an operation, and the system locks. This situation will occur randomly based on hardware interrupts emptying the prefetch during back-to-back coprocessor instructions.

This bug is in the 80386, not the math chip, and can't occur with either an 80287 or Weitek chip. Intel has been aware of this problem since Fall 1987, but the first D step chips to eliminate this bug are only now becoming available, almost a year later. Many newer machines have incorporated a hardware fix.

Some older machines are more susceptible than others. The IBM Model 80 is one of the worst. Compaq 386/16's exhibit this problem readily. Their 20 MHz machines with static ram caches are much less susceptible. Note that many 80386 EMS simulators (Compaq's CEMM) employ paging, so this problem can show up even in the real mode (actually Virtual 8086 mode). Also note that this bug is often used as a scapegoat for faulty 386 hardware and software.

Software fixes are to avoid enabling paging or not use the 80387 chip. Hardware options are to insert a PAL below the 80386 and prevent this sequence of conditions from happening. I queried several companies concerning this problem. Compaq will supply a small PAL board to 386/16 owners who have problems. Everex plans to replace the 80836 B chips with the newer D chips. I'm still waiting to hear from IBM. For those of you with clone 386 boards, two companies offer hardware fixes. Ironwood Electronics (612/431-7025 in Apple Valley, Minnesota) sells the FIX-80386 socket ($145) rated up to 20 MHz that adds less than 1/5" to the 386 profile. Ironwood is also developing a P9 board (80386 SX) for ATs. Bell Technologies (800/367-8649 in Fremont, California) sells the Math Adapter ($145) rated to 33 MHz, which uses a very small daughterboard.—*MSB* ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Table 1. Math Coprocessor Execution Times

	8231A	Intel 8087	80387	Weitek 1167	Motorola 68882
Companion CPU	8085	8086	80386	80386	68020
Typical Speed (MHz)	2	8	16	20	25
Clock Cycle time in microseconds	.5	.125	.0625	.05	.04
Single Precision (Clock Cycles)					
Add/Subtract	54-370	70-120	23-34	8-11	51-72
Multiply	146-168	90-145	29-57	7-10	59-80
Divide	154-184	190-225	88-94	32-35	69-90
Sqrt	800	180-186	122-129	117	107-126
ArcTan	6,006	250-800	191-573	157	403-422
Power	8,290-12,032	300-610	211-476	179	497-516

NOTES: Execution times are the number of clock cycles * the cycle time in microseconds. 8087 transcendental times are partial since subroutines must be employed to force arguments within restricted ranges.

in 80386 memory into the corresponding elements in WTL registers ws10 through ws29.

```
MOV ECX,20 ;load the number of elements
           in vector array
MOV  ESI,OFFSET VECTOR ;point to array
                       in memory
MOV  EDI,OFFSET WS_MULT[T10] ;point to
                     WFMUL address for ws10
REP  MOVSD ;do the multiply into WTL
           registers
```

Although the compilers available to support the Weitek WTL 1167 generate fast numeric code, hand coding crucial low level routines using these vectorizing features can yield significant speed improvements. Later, I'll illustrate the impact of hand coding on one of the benchmark programs.

Compaq and Weitek have defined a standard for user detection of the presence of the Weitek chip using BIOS INT 11h, and other manufacturers have followed Compaq's lead. The INT 11h BIOS routine (Equipment Check) returns EAX with bit 24 set to 1 if a WTL 1167 is present. If page tables have been set up so that the device is addressable from DOS real-mode, bit 23 is also set to 1. The Weitek chip can also be tested in hardware, and initializing the 1167 returns a revision number code. Wouldn't it be nice if the 80286 could be queried for its revision level?

Keep three items in mind: (1) the Weitek chip is much faster at single precision than double precision math; (2) the divide instruction takes much longer than multiply; and (3) transcendentals are slower since they must be coded as software subroutines. If all your numerics code is double precision transcendentals, using a Weitek chip will not provide as dramatic an improvement over the 80387 as it will with lots of single precision matrix manipulations.

Motorola: the "Other" Company

In the DOS world, we don't hear that much about "the other" microprocessor—the 680x0 family from Motorola. I mention it here because of the fast 68020 coprocessor boards for the PC available for the last 3 years from Definicon Systems (DSI). These boards consist of a 68020 CPU, a 6888x math coprocessor, and from 2 to 16 megabytes of memory (see *PJ* 6.1). These DSI-780 boards originally came with a 68881 math coprocessor. The MC68881 implements the IEEE standard for floating point arithmetic in hardware. It has eight floating point registers (fp0–fp7) of 80 bits (extended precision) and status and control registers. The interface between the 68020 and the 68881 is asynchronous, similar to the 80386/80387 interface so that the two chips may operate at different clock speeds. The 68881 was designed so that it could also be interfaced with other members of the 68000 family.

Last Fall, Motorola announced both their 68030 CPU and the 68882, an improved math coprocessor. The 68882 is pin compatible with both the older 68020 and the new 68030 CPUs. The 68882 is a significant improvement over the earlier 68881, both for internal execution times and for data movement instructions to and from the CPU. The 68882 can execute multiple floating point instructions concurrently. FMOVE instructions execute twice as fast as on the 68881. The 68882 was originally available as a 20 or 25 MHz part. In a bit of irony, early 68882 chips would not work with the Mac II (68020) since the 68882 would not function at the low Mac clock speed of 16 MHz. The Definicon DSI-780+ board I ran my tests on will be about two-and-a-half years old when you read this article, so it should be considered representative of Definicon's older technology. Nevertheless, the 68882 chip dropped in and worked just fine. Definicon has been shipping 25 MHz boards for some time, and they are working on even faster boards.

Give Me Them Wide Open Spaces

With the 80387 and Weitek math coprocessors, fast numerics are possible on the 80386. However, the present 640K DOS memory limit is a barrier to moving large number crunching applications to the PC. Enter the 386 DOS extenders, which I have discussed before (see *PJ* 6.2). The 386 DOS extenders set up a bridge to the DOS environment in low memory and put the 80386 chip into the protected mode so that both the larger memory space and the full 386 instruction set are available. Three 386 DOS extenders are currently available (see Figure 1), all of which now support the Weitek chip. Phar Lap and A.I. Architects (AIA) employ the same convention for mapping the Weitek chip, using segmentation to FS:0. Intelligent Graphics Corporation (IGC) implements the Unix convention, using paging.

Phar Lap 386|DOS Extender

Phar Lap markets a 386|DOS Extender (RUN386), a 386 assembler, linker, librarian, and debugger package (386|ASM), a symbolic debugger (386|DEBUG), and special tools for embedded applications (LINKLOC). Phar Lap has traditionally supplied the assembler and linker used for 386 development under DOS. Their development tools are also available for cross-development under VAX and Unix hosts. Phar Lap defined the 386 object module (EASY-OMF) most commonly used under DOS for the 386. EASY-OMF is an extension of the Intel OMF-86 object module in which some fields are extended to 32 bits for the 80386. An EASY-OMF object module is denoted by a comment record at the start of the file containing "80386". With the exception of Lahey's, all the 386 compilers used for this article generate PHAR LAP 386 object files and are sup-

Figure 1. 80386 DOS Extenders

	PharLap	*IGC*	*A.I. Architects*
Product	386|DOSExtender	X-AM	OS/386
Version	2.0	1.2	1.9.16
Price	$995	$595	$495+

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

RAMpak™ for your Compaq!

MicroWay is your best source for the software and hardware you need to get true 32-bit speed out of your 386 machine.

Since many of our users have been having difficulty obtaining memory to upgrade their DESKPROs, we decided to add a 386/20 one megabyte upgrade module to our product line. The RAMpak™ is Compaq compatible and can be used to extend your 386/20 from one to four megabytes. MicroWay's "SlotSaver" combines an mW1167 with a VGA card for the 20 MHz model. We also sell the 25 MHz Weitek 3167 for the DESKPRO 386/25.

Additionally, we have mW1167 slotless numeric coprocessor cards for Acer, ALR, AT&T, Compaq, Dell, Everex, HP, Micronics, NCR, Olivetti, Tandy and Wyse 386 systems. We will soon offer an mW1167 that runs in the Micro Channel of the IBM model 70 and 80.

Dr. Robert Atwell, a leading defense scientist, calculates that NDP Fortran-386 is currently saving him $12,000 per month in rentals of VAX hardware and software while doubling his productivity!

Fred Ziegler of AspenTech in Cambridge, Mass. reports, "I ported 900,000 lines of Fortran source in two weeks without a single problem!" AspenTech's Chemical Modeling System is in use on mainframes worldwide and is probably the largest application to ever run on an Intel processor.

Dr. Jerry Ginsberg of Georgia Tech reports, "My problems run a factor of six faster using NDP Fortran-386 on an mW1167 equipped 386/20 than they do on my MicroVAX II."

Our NDP compilers and NDP utilities are the key to taking advantage of the two to five-fold increase in speed that the mW1167 provides. The compilers generate mainframe style code, while our utilities simplify ports from either the PC or the mainframe world. A new utility enables our NDP compilers to call HALO 88.

If you are curious about the benefits of the NDP/mW1167 approach, consider the following (price estimates are for complete systems):

Coprocessor	Speed (Flops)	Prices
80287-10	80,000	$3,000
80387-20	440,000	$5,000
mW1167-20	**2,100,000**	**$6,000**

When driven by a 32-bit compiler, the mW1167 approaches the speed of a $600,000 VAX 8650! In fact, many of our users have reported increases in turnaround of 2 to 4 times that of their VAX! Call us for complete details.

32-Bit Compilers and Tools

NDP Fortran-386™ and **NDP C-386™** Compilers generate globally optimized, mainframe quality code and run in 386 protected mode under Phar Lap extended MS-DOS, UNIX, or XENIX. The memory model employed uses 2 segments, each of which can be up to 4 gigabytes. They generate code for the 80287, 80387, or mW1167. Both include high speed EGA graphics extensions written in C that perform BASIC-like screen operations.

- **NDP Fortran-386™** Full implementation of FORTRAN-77 with Berkeley 4.2, VAX/VMS and Fortran-66 extensions..... $595
- **NDP C-386™** Full implementation of AT&T's PCC with MS and ANSI extensions.... $595

NDP Package Pricing:

387FastPAK: NDP Compiler, Phar Lap and 80387 Coprocessor — 16 MHz: $1299; 20 MHz: $1499

1167FastPAK: NDP Compiler, Phar Lap and mW1167 Coprocessor — 16 MHz: $1695; 20 MHz: $2295

Phar Lap Development Tools $495
Virtual Memory Extension $295

NDP Windows™ — NDP Windows includes 80 functions that let you create, store, and recall menus and windows. It works with NDP C-386 and drives all the popular graphics adapters: ..... Library: $125, C Source: $250

NDP Plot™ — Calcomp compatible plot package that is callable from NDP Fortran. It includes drivers for the most popular plotters and printers and works with CGA, Hercules, EGA and VGA $325

NDP/FFT™ — Includes 40 fast running, hand coded algorithms for single and double dimensioned FFTs which take advantage of the 32-bit addressing of the 386 or your hard disk. Callable from NDP Fortran with mW1167 and 80387 support $250
387FFT for 16-bit compilers $250

HALO 88 to NDP Graphics Interface — This module enables you to call graphics routines in HALO 88 from NDP Fortran or C $100

MicroWay® 80386 Support (508) 746-7341

Parallel Processing

Monoputer™
The world's most popular transputer development product runs all MicroWay transputer software using either a T414 or T800. The T800 processor has built-in numerics and provides performance comparable to an 80386 running at 20 MHz with an mW1167. The new 3L Parallel C and Fortran compilers make this an especially attractive porting environment. Can be upgraded to 2 megabytes.
Monoputer with T414 (0 MB) $995
Monoputer with T800 (0 MB) $1495

Quadputer™
This board for the XT, AT, or 386 can be purchased with 2, 3 or 4 transputers and 1, 4 or 8 megabytes of memory per transputer. Two or more Quadputers can be linked together to build networks with mainframe power which use up to 100 or more transputers. One customer's application has gone from 8 hours on a mainframe to 16 minutes on a system containing five **Quadputers** ..... from $3495

Transputer Compilers and Applications
MicroWay offers Parallel languages for the Monoputer and Quadputer.
Logical Systems Parallel C $595
MicroWay Occam2 ..... $495
3L Parallel C $895
3L Parallel Fortran $895
MicroWay Prolog Interpreter $750
mField — A specialty finite element analysis package targeted at transputer networks. Ideally suited to take advantage of the six megaflop speed of the Quadputer ... $1600
T800/NAG™ (See NDP/NAG) $895

387BASIC™ — MicroWay's new 16-bit Microsoft compatible Basic compiler generates the smallest EXE files and the fastest running numeric code on the market $249

Compaq 386/20/25 Add-Ons

RAMpak™ - One megabyte 32-bit memory module fits in Compaq memory slot . . CALL
mW1167™ - Built at MicroWay using Weitek components and includes an 80387 socket.
mW1167-16 ..... $995
mW1167-20 ..... $1595
mW1167/VGA-20 "SlotSaver" $1995
Weitek 3167 25MHz $1995
80387-20 ..... $675
80387-25 ..... $795

Numeric Coprocessors

8087 ..... $99
8087-2 ..... $145
80287-8 ..... $239
80287-10 ..... $279
80387-16 ..... $450
80387-20 ..... $675
80387-25 ..... $795
287Turbo-12 (for AT compatibles) $450
256K 100ns DRAM ..... CALL
256K SIMMS $119
1 MB SIMMS $450
(All of our Intel coprocessors include 87Test.)

Intelligent Serial Controllers

MicroWay's **AT4™**, **AT8™**, and **AT16™** are the fastest 80186-based intelligent serial controllers on the market. They come with drivers for UNIX, XENIX, and PC MOS.
AT4 ... $795 **AT8** ... $995 **AT16** ... $1295

32-Bit Applications

PSTAT-386 — This mainframe statistics package has been used by government and industry for 20 years. The full version was ported. Requires 4 to 6 megabytes of memory: $1495

NDP/NAG™ — Features a library of 268 engineering and scientific numerical algorithms. Callable from NDP or 3L Fortran $895

12 MHz PC Accelerators

FastCACHE-286 12 MHz $299
SuperCACHE-286 12 MHz $399
FastCACHE-286 9 MHz $199

MicroWay

World Leader in PC Numerics

P.O. Box 79, Kingston, MA 02364 USA (508) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466
USA FAX 617-934-2414 Australia 02-439-8400 Germany 069-75-1428

Number **104** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

plied with PHAR LAP compatible libraries. Phar Lap's assembler and linker are provided in two forms under DOS: (1)a real mode version for running under DOS and (2) a protected mode version that runs under DOS on an 80386 with extended memory above 1 megabyte. The protected mode versions operate much faster, but otherwise are equivalent to the real mode versions (FASTLINK, the protected mode version, can link larger programs). 386|DEBUG is very similar to Microsoft's SYMDEB and supports symbolic debugging at the assembly language level. 80387 instructions are disassembled, but Weitek instructions are currently not supported.

RUN386 supports a number of switches and options and automatically senses the presence of other programs using extended memory (ramdisks, EMS emulators, etc). RUN386 also supports the Virtual Control Program Interface (VCPI) jointly drafted by Phar Lap and Quarterdeck Systems (DESQview) to allow multiple 386 programs to cooperate. VCPI was designed to avert the chaos that characterizes TSRs under DOS. RUN386 supports both enabling or disabling paging. With paging, RUN386 can use about 350K of low memory for 386 applications. Because of Erratum #21 (see sidebar), paging must be turned off on many 386/387 machines in order to prevent a hardware lockup. RUN386 recently added support for making calls from the protected mode to real mode, and a sample linkage to Microsoft C graphics routines is provided. Phar Lap is currently Beta testing a version of RUN386 that supports virtual memory for 386 applications. The company is aggressive about updates and software fixes.

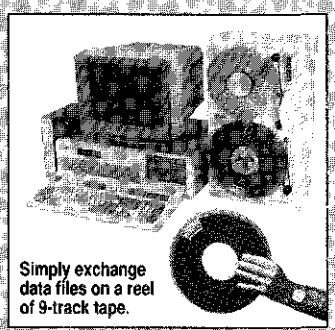
A.I. Architects OS/386

A.I. Architects (AIA) markets both hardware and software. They sell a fast 80386/80387 coprocessor board for the PC called the Hummingboard. The Hummingboard is available with up to 24 megabytes of memory at 16, 20, and 25 MHz clock rates. Gold Hill uses the AIA Hummingboard as the platform for their LISP environment. The Hummingboard uses a fast, static ram cache and can execute in parallel with the host PC machine. AIA markets two DOS extenders: (1) OS/386 designed for 386 machines and Hummingboards and (2) OS/286, a similar DOS extender designed for 80286 machines. Both operating systems set up a bridge to DOS and place the 80286 or 80386 into protected mode. The current version of OSx86 is 1.9.17, but Version 2.0 should be available by the time you read this. The current version of OS/386 does not implement paging. This decision was based partly on the Erratum #21 problem discussed in the sidebar. However, Version 2.0 will add a paging option and VCPI support.

Of the three DOS extenders, OS/386 provides the most complete emulation of MS-DOS calls. OS/386 supports a powerful Remote Procedure Call (RPC) mechanism that allows 386 code to make calls to real mode routines down in low memory. AIA supplies RPC interfaces for both GSS and Halo graphics drivers along with extensive RPC example code. This allows 386 applications to use the extensive real mode libraries until 386 versions of the libraries are available. The development software comes with a powerful command shell and debugger called CP. CP provides the equivalent of a much improved DOS COMMAND.COM with built in history, editing, and debugging commands. The shell supports a powerful macro capability such that the shell and the assembly debugger can be configured

Desktop 9-Track Tape Subsystem

Now, 9-track tape lets your micro exchange data with minis and mainframes.



Simply exchange data files on a reel of 9-track tape.

9-TRACK is the first choice for file interchange among data processing professionals. Now, Qualstar's low cost 1/2-inch 9-track *Ministreamer* tape systems bring full ANSI data interchange to IBM PCs or Macintosh, giving your micro the freedom to exchange data files with nearly any mainframe or minicomputer in the world.

Available in both 7" and 10-1/2" versions, compact Qualstar tape drives can sit on your desktop, *using less space than an ordinary sheet of paper.* Systems include DOS or XENIX compatible software, coupler card and cables. High reliability 1600 or 6250 BPI capability may be used for disk backup as well as data interchange. It's over the big advantage 9-track tape has over other micro/mainframe links.

DEC HP ALTOS NCR — IBM MAINFRAMES — AT&T UNISYS APOLLO SUN

Call us today!

FOR DETAILS AND TO ORDER:
FAX (818) 882-4081
PHONE (818) 882-5822

#1 Selling 9-Track Systems on the Desktop

QUALSTAR®

9621 Irondale Ave., Chatsworth, CA 91311

©1988 Qualstar Corp. All product and company names and trademarks are the exclusive property of their respective owners.

Number 97 on the Reader Service Card

★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The Aspen Systems Subroutine Editor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE-**

ASE is still only..... **$99**
DEMO (See what ASE can do for you!)..... **$ 3**

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

TRANSPORTABLE ★ FULLY DOCUMENTED
EFFICIENT ★ AFFORDABLE

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

ASP is still only..... **$130**

ASE and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

ASPEN SYSTEMS

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number 86 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

386 HummingBoard

TWO PROCESSORS ARE BETTER THAN ONE

Prices Start at just $1,595

When it comes to 386 add-in boards, the HummingBoard is in a class by itself. It is a true coprocessor that makes productive use of the base system's original 8086, 286 or 386, running it in *parallel* with the HummingBoard's 386.

You can adopt a whole new PC strategy.

Put your demanding applications that need more speed or memory, like 123, dBase, AutoCad, compilers, etc., on the HummingBoard while your network, Desqview, disk cache, TSR's and other applications run concurrently in the *other* 640k on the "base" system.

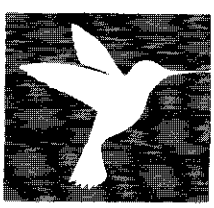
Our new Second-Shift™ software permits running a time consuming compile, report or other program totally in the background on the Hummingboard while you continue to use your base system for other tasks.

Turn 640k into 900k

Another HummingBoard benefit is more memory for DOS programs, *without changing your software.* On the HummingBoard, many DOS applications can address from 700k up to 900k directly, without EMS. EMS 4.0 is supported as well.

Blazing performance

With 16, 20 and soon 25Mhz versions, plus 32-bit cached memory for no-wait-state performance, the HummingBoard is at the top of the class for fast 386 systems.



A.I. Architects

One Kendall Square, Cambridge, MA 02139
(617) 577-8052 FAX (617) 577-9774

Number 180 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

to your own personal command set or made to emulate DEBUG or SYMDEB. AIA keeps up with updates and software fixes and maintains a technical bulletin board.

IGC X-AM DOS Extender

Intelligent Graphics (IGC) is known primarily for their VM/386 multitasking program for the 386. They also market X-AM (eXtended Address Mode), their 386 DOS extender runtime called VM/RUN. The primary differences between X-AM and the other DOS extenders is that paging must be enabled. X-AM doesn't use a segmented memory model; memory is considered as one flat memory space. DOS would be in the lower 640K, video RAM between 0A0000h and 0C0000h depending on your adapter, and 386 programs loaded at the 32 megabyte address. This requires the use of hardware paging on the 80386 to relocate physical memory to 32 MB for 386 code. One problem with paging is Erratum #21. Since paging cannot be turned off, the IGC runtime can exhibit this problem when running 386/387 code without a hardware fix. I was able to get this problem to occur even on a Compaq 386/20, which has been mostly immune because of its ram cache design. IGC follows the Unix convention and maps the Weitek 1167 to address 0FFC00000h using paging. Using the Weitek under X-AM is one way to get around the 80386/387 lockup problem.

Of the three DOS extenders, VM/RUN provides the weakest emulation of DOS. VM/RUN doesn't support a number of system calls or supports them differently than DOS does. Executable files destined for X-AM are linked using the Phar Lap linker to a relocatable REX file format. X-AM has not been updated for almost a year—IGC has been spending its resources on VM/386, its premier product. During that time, Phar Lap has added support in its linker to pack repetitive data using some new file formats; but since VM/RUN has not been updated, REX files are much larger than files running under RUN386 or OS/386. VM/RUN does not currently support the VCPI interface nor recognize when other programs are using extended memory. Alternately, running a VM/RUN application and Phar Lap's RUN386 will occasionally lock the machine. Using VM/RUN, the stack is completely protected and doesn't grow in size. A utility must be used to set the maximum stack size in executable programs. During development, the first execution of a program will typically crash with a stack protection fault until the stack size is increased. On the other hand, X-AM does have plus features. Once you buy the developer kit, you can distribute the runtime royalty free. For example, the SVS (SAIC) compilers come with free distribution rights for the supplied runtime. And I like the way VM/RUN implements executable files during development stages.

C and FORTRAN in the Fast Lane

Currently, two C compilers that support large number crunching applications on the 386 are available. Both compilers are highly optimizing and can generate inline code for either the 80287, 80387, or the Weitek 1167 math chips. An optimizing C compiler that generates inline 68881 instructions is available from Definicon for their DSI-780 board. However, C still remains a bit of an anachronism when it comes to numerics. I will describe three problems I encountered with C and number crunching.

First, C was clearly not designed for fast numerics since **float**s are traditionally coerced (promoted) to **double**s. With ANSI C and prototypes, this conversion can be controlled for user written code. But with the C math library (ANSI or Unix versions) control of conversion is problematic since all of the functions expect arguments of type **double** and return **double**s. For efficiency, you would have to rewrite the entire C math library with different names for versions of type **float**. Also, the compilers would have to be rewritten to generate inline instructions that did not promote variables from **float** to **double**. In addition, some C library routines are inherently inefficient, like the **pow()** function, which expects both the number and the exponent to be of type **double**. If the exponent is an integer, much faster code can be generated using multiplies.

Second, C supports arrays conceptually as if they were all arrays of a single dimension. When an array is passed as an argument, only a pointer to the first element is passed. A matrix (two-dimensional array) is a common occurrence in numerics code, but it is not clean or easy to write portable, efficient C functions that manipulate (say, add or multiply) matrices of variable sizes.

Third, C does not support a simple method to quickly read and write out files (often temporary files) of variables without formatting. By this I mean the equivalent of the FORTRAN unformatted reads and writes. Sure, C has **fread()** and **fwrite()**, and most compilers support the lower level **read()** and **write()** Unix system calls, but these routines read and write contiguous blocks of memory only, not collections of variables as their FORTRAN counterparts do. All of these things can be done in C, but the language itself doesn't help.

Four FORTRAN compilers currently support 386 number crunching. All four provide 80287 inline code generation, and three can generate inline 80387 code. Two compilers can generate code for the Weitek math chips. I also worked with one FORTRAN compiler on the 68020/68881. All of the compilers are ANSI FORTRAN-77 compatible. However, the problem with much existing FORTRAN source code is that it uses many early FORTRAN IV anachronisms or language extensions that are available on the forgiving IBM, VAX, and CDC mainframe FORTRAN compilers. Various VAX and IBM extensions are supported by each compiler vendor, with only one (SVS) supporting most common VAX extensions. All of the compilers support SAVEd local variables to handle older VAX or IBM mainframe code, but this typically results in a 10% speed penalty. Both the 80386 and the 68020 can access variables "register indirect" more quickly than by absolute address. FORTRAN, with all its shortcomings, continues to be the language of numerics intensive applications.

MetaWare High C-386

In addition to their fine DOS C and Pascal compilers, MetaWare markets C and

Figure 2. 80386 and 68020 DOS Compilers Reviewed

C	*MetaWare*	*MicroWay*	*Definicon*
Product	High C-386	C-386	SVS C (68020)
Version	1.4+	1.4D	2.8
Price	$895	$595	$398

FORTRAN	*Lahey*	*LPI*	*MicroWay*	*SAIC*	*Definicon*
Product	F77LEM/32	LPI-FORTRAN	FORTRAN-386	SVS FORTRAN	SVS (68020)
Version	1.0	03.00	1.4D	2.8	2.8
Price	$895	$695	$595	$895	$528

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

dp_MAX

dBase III Tools in Turbo Pascal 4.0

Complete Support for dBase III files.
DBF, NDX, DBT file & record Access.
Fully Compatible dBase III B+Tree ISAM.
Library of 100+ functions in TP4 Unit.
Allows 250+ files & indexes open at once.
LRU file caching, round robin file manager.

$$dp\_MAX = \int_{-\infty}^{+\infty} \frac{dBASE\ III +}{Turbo\ Pascal\ 4.0}$$

Max Software Consultants, Inc.
4101 Greenmount Avenue
Baltimore, MD 21218
(301)-323-5996

$149

Number 213 on the Reader Service Card

TURBO SOURCE SEARCH

TSS is a programmer's SIG that currently has 240+ disks representing 80MB of archived files!! TSS specializes in locating publicly accessible files in Turbo Pascal, BASIC, QBASIC, C, QuickC, ASM and Modula-2. We offer these files by mail as well as on-line.

TSS-BBS (617-545-9131 within the PC-Pursuit area) is an outlet for the programming code from *Programmer's Journal*. All files in the TSS SIG are freely accessible through the BBS to members and non-members. Members have increased time and access. A central library for publicly accessible programming utilities is a valuable resource, and we are certain that programmers recognizing that will support our efforts by becoming members.

TSS is also a BBS-by-mail. All files maintained by TSS-BBS are available to members for as low as $2.50/disk. All files on all disks are archived to maximize the number of bytes per disk. A yearly membership to TSS, $40.00, entitles you to purchase disks from the library for $4.00/disk. Bulk discounts can lower the price to $2.50/disk. We also offer custom filled disks. Send $4.00 for the catalog disk or $3.00 for a printed catalog. (Members receive first catalog disk FREE.)

TURBO SOURCE SEARCH
P.O. Box 11, Scituate, MA 02066
Voice: 617-545-6677
Data: 617-545-9131, 1200/2400 BAUD, 24 hrs.
VISA/MC/Check/Money Order/non-US extra

Number 35 on the Reader Service Card

Software Tools
For Programmers & Non-Programmers

★ NEW ★ VERSION **Opt-Tech Sort**™ ★ NEW ★ VERSION

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can read the data. Example: 1,000 records of 80 bytes can be read sorted and a new file written in less than 9 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or Called as a subroutine to over 30 programming languages.

Numerous features are included. A partial list of features includes: The ability to process files of any size. Many filetypes are supported including Sequential, Random, delimited, Btrieve, dBASEII & III and many others. Up to 30 key fields can be specified over 16 different types of data are supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes. **Newest Features: Faster than ever! LIM EMS support, User exits to pass or receive records. Less disk space required and much more!** MS-DOS $149.

On-Line Help™

On-Line Help allows you to easily add Help Windows to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable help systems.

The resident version allows you to add help to any system. Your Help system is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire. Upon clearing the help windows you are right back into what ever program you left.

The callable version allows you to easily display help windows from your programs. A simple Call to the help system makes the window appear. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location. MS-DOS $149. Demo $10.

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. Its a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted

To order or to receive additional information just call and receive immediate highly qualified attention!

Opt-Tech Data Processing
P.O. Box 678 / Zephyr Cove, NV 89448
(702) 588-3737

Number 16 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Pascal compilers that generate 80386 protected mode code. Actually, MetaWare has developed compilers and cross-compilers for about the widest range of computer platforms supported by a single vendor, ranging from the 8086, to Sun and other Unix workstations, the VAX, and the IBM 370. MetaWare was the first vendor to come out with a DOS 386 mode compiler. Phar Lap, A.I. Architects, and others continue to use High C-386 to write their software tools. MetaWare High C-386 was available a full year before any of its competitors (a sizable advantage), and MetaWare is the only compiler vendor that currently supports executing code under all three DOS extenders. In fact, the default 386 INIT module included with MetaWare's C-386 actually autosenses at runtime which environment it is running on and executes accordingly. However, MetaWare C must know how the Weitek chip is memory mapped (whether Phar Lap or IGC convention) since Weitek instructions are generated inline for speed. High C-386 supports numerics code generation for the 80287, 80387, and Weitek math chips. A math chip emulation library is also provided. High C, MetaWare's 8086 compiler, is the only product that currently supports using the Weitek math chip in the real mode. The 386 compiler is supplied in both a real mode DOS version that executes on PCs and a protected mode version (bound with RUN386) that executes faster on 386 machines. Phar Lap tools and distribution versions of any of the three DOS extenders are not included and must be purchased separately.

The C-386 compiler supports a raft of options and **pragma**s that control code generation, optimization, debugging, naming conventions, listings, and so on. Options may be given on the command line or incorporated into a "profile" file read at startup. The profile file may include prototypes, **define**s, in fact any legitimate C constructs, and provides a simple and elegant mechanism for compiler control. One useful option can generate prototypes for a file. A pseudo assembly language listing can also be generated (this file can't be compiled, though). MetaWare has pushed hard to make improvements and bug fixes and to track the ANSI C Standard during C-386's development. ANSI prototypes can be used to prevent the coercion of floats to doubles when making subroutine calls. Inline code is already generated for the new Weitek 3167 under **pragma** and option control. MetaWare's is the only compiler that currently supports the few additional 3167 instructions. The number of Weitek registers used for variable caching is controllable. Aside from the standard C math library, a complete set of single precision Weitek math routines with different names is also supplied.

The MetaWare compiler generates EASY-OMF format object files. The source code to the **init** and **heap** modules, along with code to part of the standard library, is provided. The compiler also comes with some valuable utilities. Probably the most useful for 386 programming is a utility that operates on EASY-OMF and Microsoft object files. BD will provide an interpreted dump of an object file, showing symbol names, comments, etc. A solid manual is provided, along with lots of readme files describing various extensions and features. From my experience, MetaWare technical support is exceptional. MetaWare compilers do have to grow on you. They aren't the easiest to use out of the box, but after you negotiate the learning curve, they are hard to part with for either DOS or the 386. MetaWare generated slightly faster code than its competitors using either the 80387 or the Weitek chip.

DEVELOPMENT TOOLS FOR GURUS, CONSULTANTS, VARS

SAVE TIME - MAKE MONEY

Our software SHELL simplifies coding & maintenance of *forms completion software*. A de facto WORKSTATION SPEC defines the SHELL structure.

- Programs delivered on four disks as X/SHELL:
  - Site licensed QBOS^ OP System
    - Hosted by CP/M or MS-DOS/UniDos
    - Interprets QBIC - a Basic dialect
  - Prototype MORTGAGE FORMS PROCESSOR.
    - Illustrates the QBOS^ USER MANUAL
    - Entree to a VERTICAL MARKET
    - Customize for other applications
      - *Billing, shipping, etc.*
  - Plain vanilla X/BOOK wordprocessor
  - Generic OP SPEC documentation:
    - 15 Pages in X/BOOK format
    - Customize for any Workstation

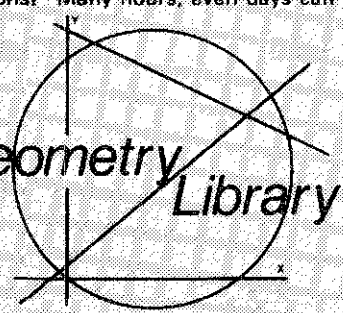
Call today for more information or
Send for $25 DEMO DISK & QBOS^ MANUAL:
Your check or we ship C.O.D.
Full refund if returned in 60 days.

X/ACT
WORKSTATION SOFTWARE
JAVERT, INC.
P.O. Box 1144 Redlands, CA 92373 • (714) 793-4915

Number **202** on the Reader Service Card

"The cost involved, in writing just one TurboGeometry routine, is more than the price of the library itself".

Are you programming or planning to program in CAD/CAM or graphics applications? Many hours, even days can be spent in writing and debugging geometric routines. *Turbo-Geometry library*



can relieve you of these time consuming tasks. There are over 150 routines in the library. Some of the areas covered are: Equations of lines, circles, and planes • Intersection of lines, circle, areas and planes • 2 and 3 dimensional transforms • Decomposition of polygon • Hidden line removal • Volume and area calculations • Perspectives • Polygon line clipping • And many more! Supported by example programs, and a 400 page manual. Source code included. 30 day guarantee. Need IBM PC/COMP, MS/PC DOS 2.0+, Turbo C, Turbo Pascal 4.0, MS C or Turbo Pascal MAC. $149.95 plus $5.00 S&H. Foreign orders add $11.00 for S&H. VISA, MC, Chk., MO. To order, call or write.

Disk Software, Inc. • 2116 E. Arapaho, #487
Richardson, TX 75081 • Tel #214-423-7288

"Give yourself a break. Get TurboGeometry Today"

Number **229** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Table 2. C Compiler Results

	-- 80387 --		-- WT1167 --		-- 68882 --	
Metaware	NDP	Metaware	NDP	SVS-20	SVS-25MHz	
FLTBENCH (2 files--1,170 lines)						
Compile and Link Times (minutes:seconds)						
Compile time	0:26	0:36	0:26	0:58	0:15	
Link time	0:05	0:05	0:06	0:05	0:04	
Total time	0:31	0:41	0:32	1:03	0:19	
File Sizes (bytes)						
OBJ	13,787	18,086	20,653	31,430	16,384	
EXE	27,580	30,612	45,611	52,604	18,944	
Benchmark times (secs)						
Float	0.77	0.60	0.44	0.44	0.93	0.77
Flt	9.39	6.26	2.31	3.02	7.36	5.93
Savage	2.86	3.19	2.09	3.02	3.19	2.58
Autodesk	4.06	4.50	2.69	3.41	4.67	3.73
w/ INTRIG	14.88	15.38	6.86	7.74	19.33	15.55
Whetstones (K/sec)						
Single	1,956	1,555	3,424	2,457	1,347	1,672
Float	1,700	1,492	2,680	2,304	1,291	1,620

NOTES: Compilers tested on Compaq 20 MHz 80386/387 and WTL 1167, also using Definicon 68020/68882 (20 MHz and 25 MHz)

MicroWay NDP C-386

MicroWay has been marketing hardware and software for numerics applications for some time. Their NDP C-386 and FORTRAN-386 compilers, however, are their first efforts in compiler development, and they'll follow soon with a Pascal-386 compiler. These compilers are available on the 386 under both Unix and the DOS extenders. The compilers are written by Green Hills (a major Unix compiler vendor), and MicroWay staff have been improving code generation, the 386-DOS supplied library, and debugging features. The compilers are globally optimizing and can require lots of memory for compilation. The C compiler is source compatible with the Unix Portable C compiler and supports features of the Berkeley C compiler. NDP C-386 supports only some of the features of ANSI C (Version 1.4d does not support ANSI prototyping, for example), and a revision is in the works to support the full ANSI C standard.

The compiler supports a large range of options for code generation, optimization, listing, etc. Inline code can be generated for 80287, 80387, and 1167 math chips. Support for the Weitek chip includes an option not to promote floats to doubles, and the math library contains Weitek routines of type float with different names similar to MetaWare's. NDP C-386 is a protected mode compiler that operates only on 80386 machines using RUN386. The compiler generates 386 assembly language files suitable for assembly with Phar Lap's 386|ASM. Weitek code can be generated either as data bytes or by using the Weitek macros. The Phar Lap tools and a distribution version of RUN386 are not included and must be purchased separately.

NDP C-386 does not require a math chip for execution; however, the code generated assumes the presence of an 80x87 or Weitek chip. Earlier versions of the compiler would run under AIA's OS/386 and on the Hummingboard. Although this latest version does not, MicroWay plans to correct this. MicroWay has worked hard to incorporate improvements and bug fixes and provides free updates for the first year. The package comes with a Mini-Make utility that can be used to set various compiler and linker options and maintain a database of **make** entries. This **make** facility also has brief online help describing the many compiler options. NDP C-386 code can be linked with NDP FORTRAN-386. The compiler library also includes a collection of graphics primitives that support CGA, Hercules, and EGA adapters modeled after the Microsoft graphics routines. The runtime library provides excellent support for numerics exceptions. It includes special C routines to interrogate and change the modes that work the same on all three possible math chips. Supporting 386/387 and Weitek library products are available for Fast Fourier Transforms (FFT) and numerical algorithms. The compiler is bundled with a solid manual with especially good documentation of numerics exception handling. Since the compiler generates assembly language files that must then be linked, compile/assemble times are noticeably longer than MetaWare's. However, if you want to bit fiddle in assembly language, the assembler files provide a unique opportunity. Execution times for numerics using the 80387 were comparable to MetaWare's with some times higher and some lower. MetaWare's C compiler generated consistently faster times for the 1167.

SVS C (Definicon 68020)

Definicon Systems (DSI) markets a family of C, Pascal, and FORTRAN compilers developed by Silicon Valley Software (SVS), another Unix and workstation compiler vendor, for their DSI-780 series 68020/68882 boards. The same basic Pascal and FORTRAN compilers are also available for the 386 from Science Applications (see below). The SVS C compiler generates inline code for the 68881 math chip. Particular attention has been applied to optimized use of floating point hardware and character manipulation. The SVS C compiler is currently compatible with the latest AT&T Unix C compiler (pcc2) and does not support the newer proposed ANSI C Standard features like prototyping. A compiler option does prevent coercing floats to doubles in expressions, but floats are passed as doubles to functions. Since the 68881 has 80-bit extended precision floating point registers, the speed penalty is somewhat lessened. The C library is a bit weak in its support of DOS features, and it doesn't support the DOS C compiler option of choosing either text or binary file types—files are always binary as in Unix. Definicon supplies source code and a linkable library of routines to interface with DOS and use special features available on the coprocessor boards.

A powerful source code symbolic debugger (SDB) is available from Definicon at additional cost. Definicon staff wrote SDB to emulate the source debuggers normally supplied with the SVS compilers. This Definicon debugger was by far the best debugger I tested with any of the 32-bit compilers. Definicon combined the symbolic source debugging features from SVS with their assembly language monitor into a powerful product. This is the only debugger that supports both high level source code (C, Pascal, and FORTRAN) and low level assembly language. Windowing is supported even for debugging. The compiler executes fast, much faster than either High C-386 or NDP C. Benchmark runtimes on the DSI 20 MHz boards are generally slower than

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the best times on the 80387. The runtimes on the 25 MHz Definicon boards are noticeably faster than on the 20 MHz DSI boards and generally slightly faster than the 80387 at 20 MHz. The 80386/Weitek times are much faster than the 68020/68882 times, which is to be expected.

SVS FORTRAN (Definicon 68020)

For the last three years, Definicon has marketed SVS FORTRAN compilers hosted on their DSI-780 boards. The Definicon SVS FORTRAN compiler generates optimized 68020 code with in-line code generated for the 68881. The latest version of FORTRAN now available on the DSI boards is the same as SVS FORTRAN for the 80386 except for the code generator and runtime libraries. Both 80386 and 68020 compilers support the same ANSI FORTRAN-77 syntax, compiler options (that aren't CPU dependent), and features. This latest version of SVS FORTRAN has the most complete implementation of VAX and IBM mainframe extensions of any of the 32-bit compilers I tested. The extensions include DOUBLE PRECISION COMPLEX data types, ENCODE/DECODE, NAME LIST I/O, inline comments, DO WHILE and END DO, extending initialization to type statements, bit fields, and arrays used as format specifiers. Also supported are Holleriths in DATA and assignment statements and in actual argument lists. Flexible parameter passing options have been added, including pass by value (for C and Pascal), as well as passing CHARACTER variables with the length vectors at the end of the argument list as in VAX FORTRAN. FORTRAN-66 DO loops are also supported. Various options can be invoked on the command line or incorporated as **pragma**s in the source code. The SDB symbolic debugger is also available at additional cost.

The current compiler release generates code that by default passes arguments using C conventions; the caller clears the arguments from the stack. This can be changed via command-line options. Although this calling mechanism is less efficient, FORTRAN code can be linked directly with SVS C code and can call routines with variable numbers of arguments as in C. All calls to low level system routines beginning with the % character are always called with the more efficient Pascal and FORTRAN mechanism; the callee clears the stack. I tested the difference between C and Pascal calling conventions and found it made less than 2% difference in speed. This would certainly depend on the source code; lots of small routines would increase this call overhead difference.

The FORTRAN compiler executes very fast. Useful warnings are issued, particularly noting type mismatches in subroutine calls, which can be helpful when porting code. Benchmark execution times on a 20 MHz DSI board were noticeably slower than the execution of SVS FORTRAN code on the 80386/387 except for one double precision benchmark. Runtimes on a 25 MHz DSI board are comparable with the 386/387 and noticeably faster for one double precision benchmark. These results are to be expected since the 68020 takes three cycles to access memory versus two cycles for the 80386. Also, the Compaq 386/20 I used for testing has a 64K static RAM cache, which allows much of the smaller test code to execute with zero wait states. The older DSI boards use conventional memory design with wait states for dynamic RAM. The 386/Weitek benchmark times speed by the DSI execution times by a factor of 2, which is to be expected.

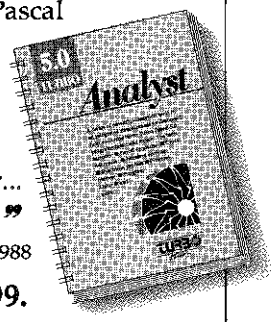
SVS FORTRAN-386

Science Applications (SAIC) markets both FORTRAN-386 and Pascal-386 compilers for the 80386 from SVS. These compilers are the same versions as those available on the Definicon DSI boards. SVS FORTRAN-386 supports the same VAX and IBM extensions as noted above. These extensions make it much easier to move older FORTRAN programs from

Write Better Turbo 4.0 5.0 Programs...

Or Your Money Back

You'll write better Turbo Pascal 5.0 programs easier and faster using the powerful analytical tools of **Turbo Analyst 5.0**. You get an integrated development environment with • Pascal Formatter • Cross Referencer • Program Indexer • Program Lister • Execution Profiler • Includes complete source code. *+ a new execution timer!*



"Turbo Analyst...a valuable tool for every Turbo Pascal programmer. The Program Structure Analyzer... a timesaver. The integrated environment...impressive."

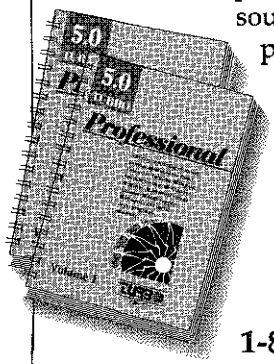
Namir Shammas, Turbo Tech Report, July/Aug. 1988

Turbo Analyst 5.0 is only $99.

Now with data entry screens!

A Library of Essential Routines

Turbo Professional 5.0 is a library of more than 500 state-of-the-art routines optimized for Turbo Pascal 5.0. It includes complete source code, comprehensive documentation, and powerful and useful demo programs. You get • TSR management • Menu, window, and data entry routines • BCD • Large arrays and more.



"Turbo Professional...a superbly crafted toolbox... [makes] the hard stuff so easy."

Kent Porter, Dr. Dobb's Journal, April 1988

Turbo Professional 5.0 is only $125.

Call toll free for credit card orders.

1-800-538-8157 ext. 830 (1-800-672-3470 ext 830 in CA)

For upgrade or other information, call 408-438-8608.

Satisfaction guaranteed or your money back within 30 days. Turbo Pascal 4.0 or 5.0 is required. Shipping & taxes prepaid in U.S. & Canada. Elsewhere add $15 per unit.

TURBO Power

TurboPower Software
P.O. Box 66747
Scotts Valley, CA 95066-0747

Circle number **25** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the VAX and IBM environments to the PC. The compiler generates inline code for the 80287, 80387, or 1167 math chips. It runs in the protected mode under IGC's X-AM 386 DOS extender. Object files in the Phar Lap EASY-OMF format are generated. The Phar Lap linker and librarian are bundled with SVS FORTRAN-386. The executable files produced must run under the X-AM runtime, which is also provided as part of the compiler package. The X-AM runtime may be distributed free without royalties. X-AM runs only with paging enabled, which can be problematic (see the Erratum #21 sidebar) on some 80386/80387 machines. This is not a problem with Weitek math chips. SVS FORTRAN allocates common blocks and variables on the stack unless told otherwise (SAVEd). The first invocation of a program will crash out of X-AM with a stack overflow until the stack size is set properly for your application.

The 386 compiler is also supplied with assembly and FORTRAN source code and a linkable library of DOS support routines written by SAIC. These routines support screen writes, character string manipulation, various DOS file operations, and access to DOS and BIOS routines from FORTRAN. An interface routine to BTRIEVE is also supplied. The SVS source code symbolic debugger is also included. Separate DBG files can be generated with a command-line option, so debugging does not require recompilation. The debugger is entirely table driven from these DBG files. The SVS debugger allows setting break and trace points, displaying source code, and displaying and setting variables and data structures. A calling sequence backtrace can display active subroutines, and memory can be displayed and changed.

The manual and supplements supplied by SAIC are adequate, but lack an index. FORTRAN-386 compiles much faster than its 386 competitors except Lahey. The benchmark execution times are also generally faster than the other 80386 compilers' using both the 80387 and 1167 except for the LINPACK benchmarks.

Table 3. FORTRAN Compile/Link Times (minutes:seconds)

	----- 80386/80387 -----				-- 68882 --
	Lahey	LPI	NDP	SVS	SVS-20Hz
LINPACK (1 file--700 lines) tests matrix math					
Compile time	0:08	0:27	0:38	0:09	0:08
Link time	0:07	0:15	0:08	0:06	0:05
Total time	0:15	0:42	0:46	0:15	0:13
File Sizes (bytes)					
OBJ	18,576	10,426	13,865	8,361	9,216
EXE	359,944	95,097	58,555	41,184	24,064
SUPERLITE (1 file--2,420 lines) tests transcendentals					
Compile time	0:43	3:40	(1)	0:38	0:37
Link time	0:09	0:20		0:15	0:08
Total time	0:52	4:00		0:53	0:45
File Sizes (bytes)					
OBJ	105,552	92,273		75,533	66,048
EXE	189,322	175,179		456,412	83,456
CHILC (3 files--1,170 lines) tests double precision					
Compile time	0:18	1:04	1:20	0:21	0:20
Link time	0:07	0:17	0:10	0:07	0:07
Total time	0:25	1:21	1:30	0:29	0:27
File Sizes (bytes)					
OBJ	43,256	52,902	43,908	29,164	25,500
EXE	141,218	118,511	77,236	68,776	46,592

NOTES: Compilers tested on Compaq 20 MHz 80386/387, also using Definicon 68020/68882 (20 MHz).

(1) Superlite contains one large procedure of 2,000 lines--NDP ran out of memory compiling on Compaq with 5 megabytes of memory.

MicroWay NDP FORTRAN-386

The MicroWay NDP FORTRAN-386 compiler is similar to their C-386 product. It was written by Green Hills, and MicroWay is enhancing and improving it for the DOS market. This compiler supports the 80287, 80387, and Weitek math chips. FORTRAN-386 is currently compatible with Unix and Berkeley f77 compilers. The package even comes with a driver program f77 that emulates the Unix f77 command options. FORTRAN-386 supports a large number of command-line options, most of them the same as on their C compiler except those that are FORTRAN language dependent. The compiler supports several VAX extensions including DO WHILE and END DO. The NDP compiler runs in the protected mode under RUN386 although previous versions also ran under OS/386. This compiler is globally optimizing, so an entire function is optimized. The compiler can require lots of memory for the tree structures and optimization tables. At a minimum, I would recommend 3 or 4 megabytes. Although it's not a problem with C, older FORTRAN code may consist of some very large procedures. I was unable to compile a 2,000 line FORTRAN procedure on a Compaq 386 with 5 megabytes of memory. The compiler generates assembly language files. These files must then be assembled and linked with the Phar Lap tools, which must be purchased separately. Because of this intermediate

Table 4. FORTRAN Compiler Results

	-------- 80387 -------				- WT1167 -		-- 68882 --		
	Lahey	LPI	NDP	SVS	NDP	SVS	SVS-20	SVS-25	VAX
Whetstones (K/sec)									
Single	1,435	1,350	1,875	2,000	3,196	4,444	1,542	1,934	1,751
Double	1,350	1,103	1,718	1,805	2,463	2,717	1,328	1,656	1,183
Linpack (MFLOPS/sec)									
Single	0.22	0.23	0.26	0.24	0.64	0.53	0.18	0.22	0.35
Double	0.15	0.18	0.21	0.19	0.36	0.36	0.15	0.19	0.14
Runtimes (min:secs)									
Superlite	2:41	3:13	(1)	2:30	(1)	1:05	3:19	2:40	1:53
Chilc	7:16	10:21	7:30	7:09	4:53	3:55	7:10	5:39	8:15

NOTES: Runtimes on Compaq 386/20 with 80387 and WTL 1167, also using Definicon 68020/68882 (20 MHz and 25 MHz); VAX 11/785+ with FPA.

(1) Superlite contains one large procedure of 2,000 lines--NDP ran out of memory compiling with 5 megabytes of memory.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

People are talking about us.

"This is a very good implementation of FORTRAN, better than the one I use on my minicomputer system." *IEEE Software*

"Lahey F77L is definitely for the programmer's market, with features for the casual and professional user...*EDITOR'S CHOICE" PC Magazine*

"It is a fantastic product." *PC Australia*

"It should be the last FORTRAN compiler you will need to buy..." *Your Computer*

"Lahey F77L was by far the easiest compiler to use..." *Programmer's Journal*

"Compilation speed is Lahey FORTRAN's most remarkable feature; it is unbelievably fast...Realistically, Lahey FORTRAN is the most efficient and productive FORTRAN development tool for the DOS environment..." *Byte*

When people talk about FORTRAN the name mentioned most often is Lahey Computer Systems Inc.

Contact us to discuss our products and your needs. (800) 548-4778
Lahey Computer Systems, Inc. P.O. Box 6091, Incline Village, NV 89450
Tel: (702) 831-2500 FAX: (702) 831-8123 Tlx: 9102401256

Circle number 44 on the Reader Service Card

The Heap Expander™ version 2.0

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

still $59.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*Idaho residents add 5% sales tax
Foreign customers add $4.00 for shipping and handling.

The Tool Makers
75 Mt. Hermon Rd.
Scotts Valley, CA 95066
(408) 485-0690

Circle number 166 on the Reader Service Card

step, NDP compile/assemble times are slower than SVS and Lahey compilers'.

NDP FORTRAN provides one feature not present in any of the other compilers. It figures out at compile time which variables must be SAVEd and converts these to statics. The compiler can also check FORMAT statements at compile time. NDP FORTRAN can be transparently linked with NDP C code. MicroWay stays on top of bug fixes and improvements. Most recently, they have rewritten the underlying file I/O routines to speed unformatted reads and writes. The next update is to add most common VAX FORTRAN extensions. The package comes with a Mini-Make facility, 386 library routines for windowing and graphics, debug routines, and some other utilities. Separate linkable 386 libraries are available for FFTs, scientific numerical algorithms (NAGs), and plotting. The thorough documentation contains excellent sections on numerics and optimization. The numerics code generated executes fast, the fastest times of the compilers for the LINPACK benchmarks.

LPI FORTRAN

Language Processors (LPI), another Unix compiler vendor, has ported FORTRAN, COBOL, PL/1, BASIC, C, and RPG compilers to the 80386 under Unix and Xenix. Now most of these languages (except C and RPG) are also available from LPI under the 386 DOS extenders. LPI FORTRAN supports both 80287 and 80387 inline code generation, and they are working on Weitek support. The compiler runs in the protected mode under RUN386. The Phar Lap libraries and linker are bundled with the FORTRAN package. The compiler can generate code to run under either the Phar Lap or IGC runtime. Their **init** module senses at runtime which DOS extender is operating. The compiler supports a variety of options: control over optimization level, FORTRAN language extensions, listing and cross reference control, and inline math code generation. The compiler can also generate a pseudo assembly language listing using Unix, not Intel, mnemonics. The package comes with a driver program and a post processor to build LNK command files, so an entire program in several hundred modules can be built with one command.

One of LPI's advantages is that LPI FORTRAN can be linked with any of their other languages. The documentation is thorough. The compiler executes more slowly than SVS's and Lahey's, comparably to NDP. I had to turn down the op-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

timization level to get most of my benchmarking programs to execute properly. The code generated for the benchmark programs generally executed more slowly than the other FORTRAN compilers'.

Lahey F77LEM/32

Lahey's is the most recent 80386 FORTRAN compiler, released this September. Lahey offers a full range of DOS FORTRAN compilers including: (1) LP77, Lahey Personal FORTRAN 77, an inexpensive yet full implementation for learning (separate code and data segments limited to 64K); (2) F77L, the fullblown DOS compiler; (3) F77LEM, their DOS compiler that generates code to run under AIA's OS/286 on 80286 or 80386 machines, removing the 640K barrier for large applications; and (4) F77LEM/32, the true 32-bit compiler that generates 80386 code to run under AIA's OS/386. All four compilers support the same ANSI FORTRAN-77 syntax and limited IBM and VAX extensions including NAMELIST. F77LEM/32 generates 80386 code and inline 80287 numerics instructions. Intrinsic functions supplied in the library sense at runtime and will use 80387 instructions for transcendentals if available. Lahey is working on adding Weitek chip support. The compiler features a variety of options that may be taken from a F77L3.FIG config file for convenience or be overridden from the command line. Another useful feature is that the compiler reads a FIX file at startup with brief text commands and then self-patches itself at runtime. This allows you to get a short bug fix for a problem over the phone or via their bulletin board and be up and running again.

The compiler generates object files based on Microsoft MASM 5.1 extensions to the Intel OMF-86 format. Lahey supplies their own 386 linker (L32) for linking these files since the Microsoft Linker cannot link true 32-bit object modules (can't exceed the 64K segment barrier). Lahey also provides a library manager (LM) for manipulating 32- bit libraries. These tools are required since Lahey is the only vendor using this object module format for 80386 code. The other 386 DOS vendors are using Phar Lap's EASY-OMF format and software tools. The executable files created will run under AIA's OS/386 DOS extender on 80386 machines and on AIA Hummingboards. The OS/386 runtime is available for distribution at additional cost.

The Lahey family of FORTRAN compilers are all easy to use. The compilers issue good diagnostics and useful warnings for such things as variables never used and unreachable code. A competent source code FORTRAN debugger (SOLD) is included. The compilers can generate symbol information stored in a separate SLD file, allowing symbolic debugging without recompilation. The SOLD debugger is similar to the SVS debugger, but with fewer features. One advantage of F77LEM/32 is that the same source code should compile with their F77LEM (16-bit) compiler and run under the OS/286 DOS extender on AT class machines with extended memory. The compiler comes with a supplemental disk of example code to handle numeric exceptions and show how to use the AIA RPC mechanism to link with real mode code. An interface to GSS is provided via the RPC mechanism along with RPC interface code that may be called directly from FORTRAN. Lahey supports a long list of compatible library products although most are real mode 8086 code. The manual is excellent, fully documenting the FORTRAN syntax supported and with appendices that describe compiler options, linking to assembly language, compiler and runtime limitations, numeric

MetaWINDOW - Power Graphics for your PC!

PC TECH JOURNAL
"Product of the Month"
"... a technological tour de force for fast PC graphics."

NO ROYALTIES!

COMPUTER LANGUAGE
Oct '87 - Product Review
"...surprisingly powerful for the cost!"
Independently benchmarked the fastest of all graphic systems reviewed - faster than Halo, over 3 times faster than GSS*CGI!

MetaWINDOW is the advanced, high performance graphics development toolkit which bridges the gap between low-level graphic primitive libraries and pre-packaged window managers. MetaWINDOW includes an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.

"MetaWINDOW is the best graphics package I've seen. Since I bought my PC, I've spent over $2500 on software. I've bought bad software, mediocre software, and good software. There's not much software I call excellent, but MetaWINDOW is excellent!"
J. Kayser
CMSI
Portland, OR

Unmatched Features!
- Display multiple bitmap or "filled-outline" fonts.
- Face fonts for bold, italic, underline or strike-out stylings.
- Full 8 function "rasterOp" transfers for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
- Create pop-up menus, windows and icons.
- Supports 70+ graphics displays including PS/2 VGA & MCGA.

- auto-cursor tracking
- pull-down menus
- pop-up windows
- comprehensive graphic functions
- unlimited fonts

File
New ^N
Open
Close
Save ^S
Quit ^Q

10 Point 12 Point
Bold Italic

MetaWINDOW comes complete with language bindings for 20 popular C, Pascal and Fortran compilers, plus dynamic run-time support for over 70 graphics adaptors and input devices.

MetaWINDOW
Advanced Graphics Development Toolkit
4 disks, 3 260 page manuals - $195*

New - MetaWINDOW/Premium
Power graphics for new high performance TI34010 based graphics cards! -$495*
(Licensing required for redistribution of MetaWINDOW/Premium)
* Plus $5.00 shipping and handling

TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

Circle number 143 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

vLIB

the user interface library for C

vLIB is a comprehensive, easy to use library of over 175 custom C functions for building sophisticated PC applications.

Windows • *overlapping, tiled, built-in window management*

Menus • *vertical, horizontal pulldown, popup, arrays*

Forms • *full screen or popup data entry with multiple field types*

Pop Ups • *messages, prompts, selection lists, scroll bars*

Formatted Screen Output

Full Editing Input

High Speed Display

... and much, much more!

Give your programs a professional look! Develop applications faster!

Compilers supported: Microsoft C and Quick C, Borland Turbo C, Lattice C, and Datalight C.

All memory models supported.

Demo disk with usable sample programs and source (includes shipping) . **$5**

Library with reference card and 280 page programmer's manual **$99**

Library (as above) plus full source code **$149**

$5 shipping and handling. California residents please add 7% sales tax.

No royalties. Site license available.

Call or write:

Pathfinder Associates
291 Madrone Avenue
Santa Clara, CA 95051
(408) 984-2256

Visa and MasterCard accepted.

BBS: (408) 246-0164 (1200/2400 N-8-1)

Circle number 145 on the Reader Service Card

exceptions, and utility library calls. All extensions to ANSI FORTRAN-77 are denoted by shaded text. The manual index alone is 30 pages. Lahey provides a technical bulletin board for downloading FIX files for the compiler and uploading bug reports and messages. Compile times are very fast. Benchmark runtimes for 80x37 numerics code were mixed, with most in the mid-range of execution times.

Benchmarking

For testing purposes, I ran compiles and execution benchmarks in the same hardware—a Compaq 386/20 with 5 megabytes of 32-bit memory and a 150 Megabyte ESDI hard disk. I used a MicroWay mW1167/20 board for testing Weitek execution. A Definicon DSI-780+ board (20 MHz 68020/68882) with 2 megabytes of memory was also tested in the same environment although I normally run it in an old PC. I also used an AIA Hummingboard (20 MHz 80386/80387) for testing, but the 387 results were generally similar to the Compaq results, so for space reasons I've left them out. However, I also received a Weitek 3167 chip, which I tested in a new Compaq 386/25, and I will report these times. For mainframe comparison, I give CPU execution times (not wall time) that I ran on a VAX 785 with a floating point accelerator.

I used a series of numerics intensive programs for benchmarking. The C programs are clearly the weakest. Most are way too short programs from *BYTE* (July 1987, p.101). I also used a transcendental benchmark from Autodesk (the AutoCAD folks) and a hand translation of the Whetstone benchmark by Tom Pennello of MetaWare. (Does anybody out there have some large number crunching programs written in C?)

The FORTRAN benchmarks are similar to programs I have used in previous articles (see *PJ* 6.1 and 6.2). However, I used the most recent LINPACK source code from Argonne Labs and made a small correction to the Whetstone source code. I don't put much value on the Whetstone benchmark. Whetstone specific optimizations have been incorporated in many PC and workstation compilers that don't affect the execution of real programs. The LINPACK programs, however, are good measurements of well written code executing matrix manipulations. The Superlite benchmark is a ray tracing routine that calculates daylight in a space—all single precision with lots of transcendentals. The CHILC program is all double precision with few transcendentals. My conclusion based on these last three programs is that a fast 20 MHz 80386/80387 or a 25 MHz 68020/68882 appears almost competitive with VAX 785 execution times. For double precision, the microprocessors seem to be superior.

Compared with the 80387, the Weitek 1167 chip in a 20 MHz 80386 makes a dramatic improvement in benchmark execution times, ranging from 1.8 to 2.3 times faster. Single precision programs show the greatest improvement with the 1167, which is to be expected. A 20 MHz 80386/1167 clearly betters the VAX times for these benchmarks. Stepping up to a 25 MHz 80386/3167 improves execution speed by from 22 to 43%; code with transcendentals or double precision showing the least improvement. Now our 25 MHz PC crunches numbers at twice the speed of the VAX. Handcoding some low level routines using the vector properties of the Weitek 1167 (HLINPKU) showed another 80% speed improvement.

How much of this speed improvement will be experienced with large applications is still conjecture. VAX minicomputers and mainframes have sophisticated I/O processors that off load tasks in parallel from the central CPU. The PC generally doesn't support parallel processing and specialized I/O processors. Many numerics applications may also do lots of file I/O to store temporary calculations. My last benchmark example is a real program, the LOADS simulation from

Table 5. Numeric Benchmarks--Weitek 3167 (25 MHz)

	C		FORTRAN		
	MetaWare	NDP	NDP	SVS	VAX
Whetstones (K/sec)					
Single	4,545	3,246	4,032	5,378	1,751
Double	3,571	3,030	3,030	3,312	1,183
Linpack (MFLOPS/sec)					
Single			0.82	0.76	0.35
Double			0.46	0.46	0.14
Linpack (MFLOPS/sec) -- handcoded one routine					
Single				1.37	
Double				0.56	

NOTE: Runtimes on Compaq 386/25 with WTL 3167; VAX 11/785+ w/FPA

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Table 6. Numeric Benchmarks--DOE2 LOADS module

	SVS FORTRAN 2.8			VMS 4.5
	80386/1167	80386/80837	68020/68882	VAX
Runtimes (min:secs)				
LOADS	3:07	4:11	5:18	2:52

NOTE: Runtimes on Compaq 386/20MHz using 68020/68882-20 MHz; VAX 11/785+ w/FPA.

DOE2, a large building energy analysis program developed at Lawrence Berkeley Labs. The LOADS module in DOE-2 is about 14,000 lines of code. I used SVS FORTRAN-386 since it was the only compiler that would swallow the source code (VAX extensions). Using a small input file, this simulation run created about a megabyte of temporary files passed to subsequent modules. The 20 MHz 80386/1167 is competitive, executing 9% more slowly than the VAX. Hand coding the most frequently used numerics routines for the 1167 would probably be very worthwhile. The impact of the math chip can be seen in the 80387 runtimes, which are another third slower. You need to work on file I/O as well as use new math chips in order to achieve substantial improvements on the PC.

A final exciting late development is a recent conversation with a *PJ* reader. Using a Definicon DSI board in an 80386 computer, he has implemented parallel processing. What once took two hours to process in sequential fashion as two operations is now implemented and running in parallel on the 68020 and the 80386 and requires only an hour. This could be done with an AIA Hummingboard. Splitting up a task for parallel execution could possibly eliminate lots of file I/O and provide another speed boost, too. ♦

WTL 1167 & 3167
Weitek
1060 E. Arques
Sunnyvale, CA 94086
(408) 738-8400

386|ASM/LINK & 386|DOS Extender
Phar Lap Software
60 Aberdeen Avenue
Cambridge, MA 02138
(617) 661-1510

OS/386 Developer's Kit
A.I.A. Hummingboards
A.I. Architects, Inc.
One Kendall Square
Cambridge, MA 02139
(617) 577-8052

X-AM Development Environment
Intelligent Graphics Corporation
4800 Great America Parkway
Santa Clara, CA 95054
(408) 986-1431

High C-386
MetaWare Inc.
903 Pacific Avenue, Suite 201
Santa Cruz, CA 95060
(408) 429-6382

NDP C-386 and FORTRAN-386
mw1167 Weitek Board
MicroWay
P.O. Box 79
Kingston, MA 02364
(508) 746-7341

DSI 780 Series Boards
SVS C, SVS FORTRAN,
and SDB Debugger
Definicon Systems
1100 Business Center Circle
Newbury Park, CA 91320
(805) 499-0652

F77LEM/32
Lahey Computer Systems
917 Tahoe Blvd, Suite 203
P.O. Box 6091
Incline Village, NV 89450
(800) 548-4778

LPI FORTRAN
Language Processors
400-1 Totten Pond Road
Waltham, MA 02154
(617) 890-1155

SVS FORTRAN-386
Science Applications
5150 El Camino Real, Suite C-31
Los Altos, CA 94022
(415) 960-5931

I'm grateful to Compaq Computer Corp. for the loan of the Compaq 386/20 machine I used for testing. During the testing, this machine performed flawlessly. I would also like to thank MicroWay for the loan of several mW1167 boards for testing, A.I. Architects for the use of a Hummingboard, and Motorola for a 68882 chip. —MSB

Form-Fill Toolkit

Product: C Library
Features:

Price: $239.95
Includes Source Code

Cursor Control
Sound & Printer Control
Fast Screen Access
Keyboard Access
Edited Field Entry
Pop-Up Window Support
Display Text Windows
Menu & List Selections
Form-fill Routines

Compiler Support
Microsoft 4.0, : Y
Microsoft 5.0 : Y
Quick C : Y
Turbo C : Y
Other:

↑↓ Scroll F1 Help F2 Save F4 Cancel Esc No Save

Provide a consistent user interface for your programs with over 100 documented functions.
SPECIAL! Includes powerful multi-record data collection module with fully commented pristine source code.

Anji Software, Inc.
P.O. Box 297
Cedar Grove, NJ 07009
(201) 239-4328

- **Ask for our Demo Diskette**
- Specify compiler when ordering
- No royalties required
- **30 Day Money Back Guarantee**

We accept Visa and M/C
NY & NJ residents add sales tax
Shipping: $3.00 Domestic
$10.00 Foreign

Circle number 162 on the Reader Service Card

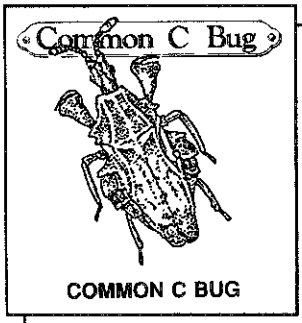
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

T-SHIRTS

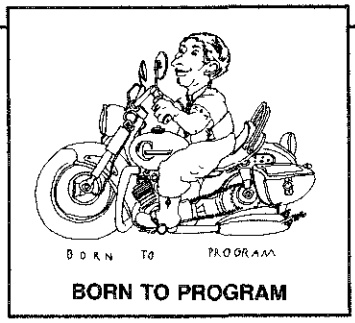
Multi-color imprints on the highest quality 50/50 cotton T-shirts.

$12.00 each — includes shipping & handling

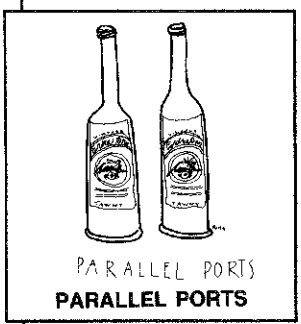
SIZES S, M, L, XL



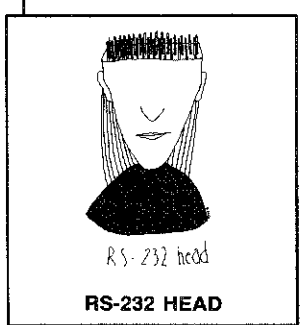
COMMON C BUG



BORN TO PROGRAM



PARALLEL PORTS



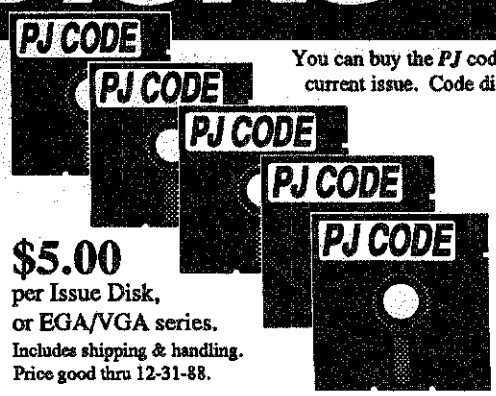
RS-232 HEAD

BUGS RULE 4 color imprint on a white shirt

All mail orders must be accompanied by check or Money Order — PREPAID ONLY.

CODE DISKS

Use the order form on the card opposite.
Order by phone with your VISA/MC.
CALL TOLL FREE
800/234-0386 ext 53



$5.00 per Issue Disk, or EGA/VGA series.
Includes shipping & handling.
Price good thru 12-31-88.

You can buy the *PJ* code listings from volume 5.5 to the current issue. Code disks for the Michael Abrash EGA/VGA series are also available.

Diskettes are available for the following issues:	
5.5	6.4
5.6	6.5
6.1	6.6
6.2	EGA
6.3	VGA

360 K MS-DOS disks standard, other formats available

PJ ISSUE SPECIAL!

CODE DISKS $5.00 REG $10.00 each

PJ Code on Disk. . .

To celebrate our new ***PJ Code DiskScription*** program (see page 8) the ***PJ* Issue Special** this time is an **incredible offer** on back issue copies of ***PJ*** code disks! Any of the issue code disks, the VGA or EGA disks—any of the code you may have missed—**only $5.00 each!** That's a savings of **50%** off the regular price! Get ***PJ*** code listings into your computer at fantastic savings—and while you're at it, start your charter ***PJ Code DiskScription***!

BACK ISSUES

$7.00 each PREPAID ONLY

A few of the Back Issues currently available

Vol. 4.6: DTACK GROUNDED; Unix Books, Part 1; How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate ans Stay Resident; Turbo Pascal inline code; Pascal—Assembler souce code for DOS function calls; Alignment problems in C and Managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laser printer; GenieferdBASE applications generator reviewed; Advanced Trace86 debugger reviewed.

Vol. 5.2: DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C—portability in practice and using header files; Proposed ANSI C Language Standard, Part 6; Indexing in BASIC; A first look at Turbo C.

Vol. 5.3: Starting your own software house--publicity; Unix books, Part 2; Consulting—merger mania; EGA data control; Modula 2 for Pascal lovers; Optimizing Turbo Pascal; ANSI C language standard; Coping with ANSI C and forcing file updates; Formatting BASIC output with buffered PRINT USING.

Vol. 5.4: Starting your own software house—technical support; Unix books, Part 3; Consultant productivity and effectiveness; EGA set/reset capabilities; ANSI C language standard; Character tables and string functions in C; MWWIND—a menu with windows utility; Variable passing and hiding in QuickBASIC 2 subprograms.

Vol. 5.5: Intel's 80386 vs. OS/2; Metagraphics President Jack Davis interview; Consulting—making computer tools pay for themselves; ASM—the invisible CapsLock; TSRs using expanded memory with Turbo Pascal; Note on optimizing Turbo Pascal; Proposed ANSI C Language Standard; Printer output from Turbo C programs; Matchup of Turbo Basic 1.0 and QuickBASIC 3.0.

Vol. 5.6: Multics, OS/2, paging and segmentation; The Usenix conference and other news; How to get rich in consulting; Shell in Marshal Pascal; Turbo Pascal 4.0 preview; Handling fixed length character arrays in C; The proposed ANSI C Language Standard.

Vol. 6.1: Consulting is an attitude; Write mode 3 of the VGA; The LIM 4.0 expanded memory specification; Environment variables in Turbo Pascal 4.0; A timer program in C; Porting mainframe software to the PC--FORTRAN compilers under DOS and the Defincon 32-bit coprocessing engines.

Vol. 6.2: Hal Hardenbergh on RISCs, CISCs, and Fabs; Pink noise; Yet another VGA write mode; Inside Turbo Pascal 4.0; The proposed ANSI C Language Standard; Converting a Turbo C program to a TSR; 80286 and 80386 programming; 80386 software development; The Norton Guide for BIOS/DOS Assembly.

Vol. 6.3: Hal Hardenbergh on memory trends; Pink noise; Getting your consulting tools in order; Gordon Letwin on OS/2 memory management; Virtualizing the 386; Pascal—A CASE of the jumps; Turbo Professional 4.0; BASIC—CALL, GOSUB, and multi-line functions

Vol. 6.4: Job-shopping; Windowing systems overview; Reading VGA memory; X Window—a view from DOS; A digital clock in C for MS Windows; Windowing in TopView, DESQview, & MS Windows; MS C 5.1 and the ANSI Standard; A mouse unit for Turbo Pascal 4.0.

Vol. 6.5: Hal Hardenbergh on computers in the fast lane; Consulting—no-win jobs and too much work; VGA text paging; Using dynamic arrays and records in Turbo Pascal 4.0; An OS/2 Presentation Manager application template; Watcom C 6.0 and the ANSI Standard; Object oriented OS/2 in C++; Ada and software engineering; REXX on the PC.

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Tools

from The Software Family

C

Borland Turbo C v2	New!	$99
Borland Turbo C v2 Prof.	New!	$169
Blaise C Tools Plus/5.0		$95
Blaise Turbo C Tools		$95
Blaise Power Screen		$95
Creative Vitamin C		$159
Creative VC Screen		$129
Essential Graphics v2		$232
Essential Screen Star w/src		$169
Gimpel PC Lint		$99
Microsoft C v5.1		$279
Microsoft Quick C		$68
Novell Btrieve		$184
Novell XQL		$599
Oakland Group C-Scape		$285
Zortech C++		$125

Object Oriented

Digitalk Smalltalk/286	New!	$164
Guideline C++		$269
Ithaca HOOPS		$455
Object Oriented Graphics Database		
Whitewater ACTOR		$449
Zortech C++		$125
Optimizing C++ Compiler		

Other Languages & Tools

Borland Assembler & Debug	New!	$99
Borland Turbo Pascal v5	New!	$99
Borland Turbo Pascal Prof.	New!	$169
Matrix Layout		$129
Microsoft Macro Assembler		$98
Microsoft Windows SDK v2.1		$328
Mortice Kern MKS Toolkit v2.3		$145
Periscope I 512K v4	New!	$579
Periscope III v4		$1079

Free Catalog & Newsletter

We carry hundreds of products from dozens of publishers. Call or write for your free subscription to our catalog and newsletter.

Returns

The Software Family wants you to pay only for the tools you really use. If you order anything featured in this ad (or most of the products in our catalog) you can open and use the product for up to 30 days and still return it if it doesn't meet your needs.

Multi-Tasking

DesqView *by Quarterdeck* is the only multi-tasking environment that provides complete compatibility for the entire PC hardware family while fully exploiting the the advanced features of the 80286 and 80386. **$90**

API Reference **$54**

API Ref & C Library **$179**

SQL

XQL *by Novell* interface to Btrieve supports both embedded SQL for programs and interactive SQL for users. XQL provides sophisticated data manipulation features which allow you to access data by field name, move forwards and backwards through the database, compute fields, and even work with composite records from joined files. **$599**

Assembler Analysis

ASMFLOW *by Quantum Software* analyzes MASM source to generate flow charts, tree diagrams, stack sizing estimates, register usage analysis, X-ref and more! New! **$79**

Debugging

New version! **Periscope** *v4 by Periscope*: The Periscope debuggers provide the combinations of hardware and software you need to solve really tough programming problems. Periscope includes all the debugging features you need, but with minimum overhead and maximum flexibility. With Periscope you can debug programs which need all memory, have real-time requirements and/or are picky about vector usage.

Periscope I 512K **$579**
requires 0k main memory

Periscope III **$1099**
hardware breakpoints and trace-back for real-time applications

The Software Family

800-443-7176

(415)957-0111

Visa • MasterCard
COD • Check

- Most orders shipped within 24 hours
- Free shipping on orders over $100

The Software Family
575 Mission Street, #600-PJ
San Francisco, CA 94105

Visit Our Store!

The Software Engineering Store

575 Mission Street, 6th Floor
San Francisco
Mon-Fri 10-5, Sat 10-3

Technical Support

The Software Family carefully evaluates features, quality and support before offering a product to our customers. All of the products featured in this ad are backed by excellent technical support from the publisher — in many cases directly from the program developers. All are published by companies with reasonable update policies and with a track record for responding to customer requests for both bug fixes and new features. We at The Software Family further back the products with our own technical support. We are willing and able to intercede with the publisher if your needs aren't satisfied or to research problems caused by the interaction of products we recommend. We can also provide quotes for custom modification of source-licensed products, in-depth technical consultation or custom-developed products.

Prices subject to change without notice - ©1988 TSF All Rights Reserved

Number 98 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Make Utilities

PolyMake
Polytron Corporation
1700 NW 167th Place
Portland, Oregon 97006
(800) 547-4000

One of the more useful tools available under Unix is the make utility. Based on a dependency file and the dates of the modules, make automates the building or updating of a program from several modules. Although several popular DOS compilers (Microsoft and Borland) come with make utilities, these are rather modest implementations with significant limitations. Polytron has released Version 2.1 of PolyMake ($149), a full featured make utility for DOS. The program is also available under VAX/VMS and as a networking implementation for PCs. PolyMake emulates the Unix make facility, in both command-line operation and dependency file syntax, but adds significant extensions and new features. PolyMake supports 20 different command-line options to control operation, including the ability to use a file for builtin commands or to disable builtins altogether, to create a batch file for execution, to give environment macro variables precedent, and to print out the makefile with macros expanded. There is even a debug/trace mode.

The dependency file syntax has also been extended, with powerful conditional constructs similar to the C preprocessor's. As in C syntax, left and right braces may be used to group operations that should be performed together. A rich set of predefined macros is provided. PolyMake supports two special targets, .INIT and .DEINIT, to be executed on initialization and after the make process has been completed. This feature can be especially helpful in setting up an environment for different build processes and restoring the state at the process end. Dependency types have been extended to include a library member reference. This allows makefiles to express a relationship between executables and specific library object modules, and not be affected by other members' being added to a library. A local input script feature allows getting around the 127 character DOS limit for commands, particularly for linking. For example, when a link operation is executed, the local input script is copied to a temporary file and the following command is executed: LINK @tempfile. The @ character passed to the operation can be changed to any printable character.

One other strength of PolyMake is that it can be directly linked with PVCS, Polytron's source code revision system. Polytron is currently working on versions of PolyMake for Unix and OS/2. Documentation for PolyMake consists of a manual for Version 1.0 (dated 1984), plus a supplement for Version 2.0, totaling about 90 pages. Information on new Version 2.1 features is documented in various read.me files on disk. No index is provided. Clearly, the manual could be improved. PolyMake uses about 120K of memory for operation. Using PolyMake, I was able to execute some sample Unix makefiles without difficulty.

Seidel Make and SMKgen
Seidel Computer Engineering
3106 Hilltop Drive
Ann Arbor, Michigan 48103
(313) 662-8086

While the make facility concept was defined in Unix, companies such as Seidel have greatly improved the basic design. Seidel has released Version 2.2 of the Seidel Make utility ($99.95) along with Version 2.0 of

MISOSYS

Programmer's Journal downloads are now available from MISOSYS' forum on Compuserve: GO PCS49

DED86™ Version 2

Powerful features in Version 2!

When you need to travel through your disk drive, why settle for a tool that isolates you from the road? DED86 gives you the direct controls you need to explore your disk. It's a full-screen sector-oriented disk/file editor and a page-oriented memory editor. When you want to "unerase" erased files, DED86's flexible KEEP facility does the job without you fussing over FATs.

- Look by cylinder/head/sector, sector or cluster
- Scan free clusters to search for erased data
- Jump about subdirectories
- Keep sectors & clusters for writing to a file
- Edit bytes in hexadecimal or ASCII, zap in 0s
- Search disk, file, or memory in ASCII or hex
- Touch directory with your date and/or time
- Obtain complete disk statistics in one screen
- Alter attributes: archive, system, hidden, read
- Save/Restore sectors to/from auxiliary buffers
- DOS subshell available while using DED86
- Handles 5.25" & 3.5" drives & RAM disks

RATFOR-86™ 2.0

If you're still fussing with FORTRAN, RATFOR-86 by James Beard, Ph.D., provides structure and greater portability to FORTRAN programs; reduce your programming time and effort dramatically over that required when writing directly in FORTRAN.

- Has extensive macro support
- "arith" macro for binary arithmetic
- 6 new conditional macros in 2.0
- short form READ and PRINT
- keyword "cswitch" like in C
- Requires a FORTRAN compiler

ED/ASM-86™

Integrated Assembler

Editor, assembler, linker, and debugger are integrated into a single .EXE file to ease you into assembly language. Handles 8086, 80186, 80286, and 8087.

- A full featured line EDITOR with intra-line editing, block move/copy, partial save/load.
- A full-screen "Turbo-type" editor included
- ASSEMBLER supports structured coding, can directly generate .COM or .EXE file; assemble to memory for faster debugging.
- The DEBUGGER supports symbolic disassembly of arbitrary code to disk

Purchase now and get a free update to **ED/ASM-386** available soon.

DED86 . . . . . .$59.95
ED/ASM-86 . . . . . .$99.95
RATFOR-86 . . . . . .$99.95
M/C and VISA accepted
S&H: $5, $6 Canada, $15 Other

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
800-MISOSYS or 703-450-4181

Number 121 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TURNS LITTLE LEAGUE into GIANTS

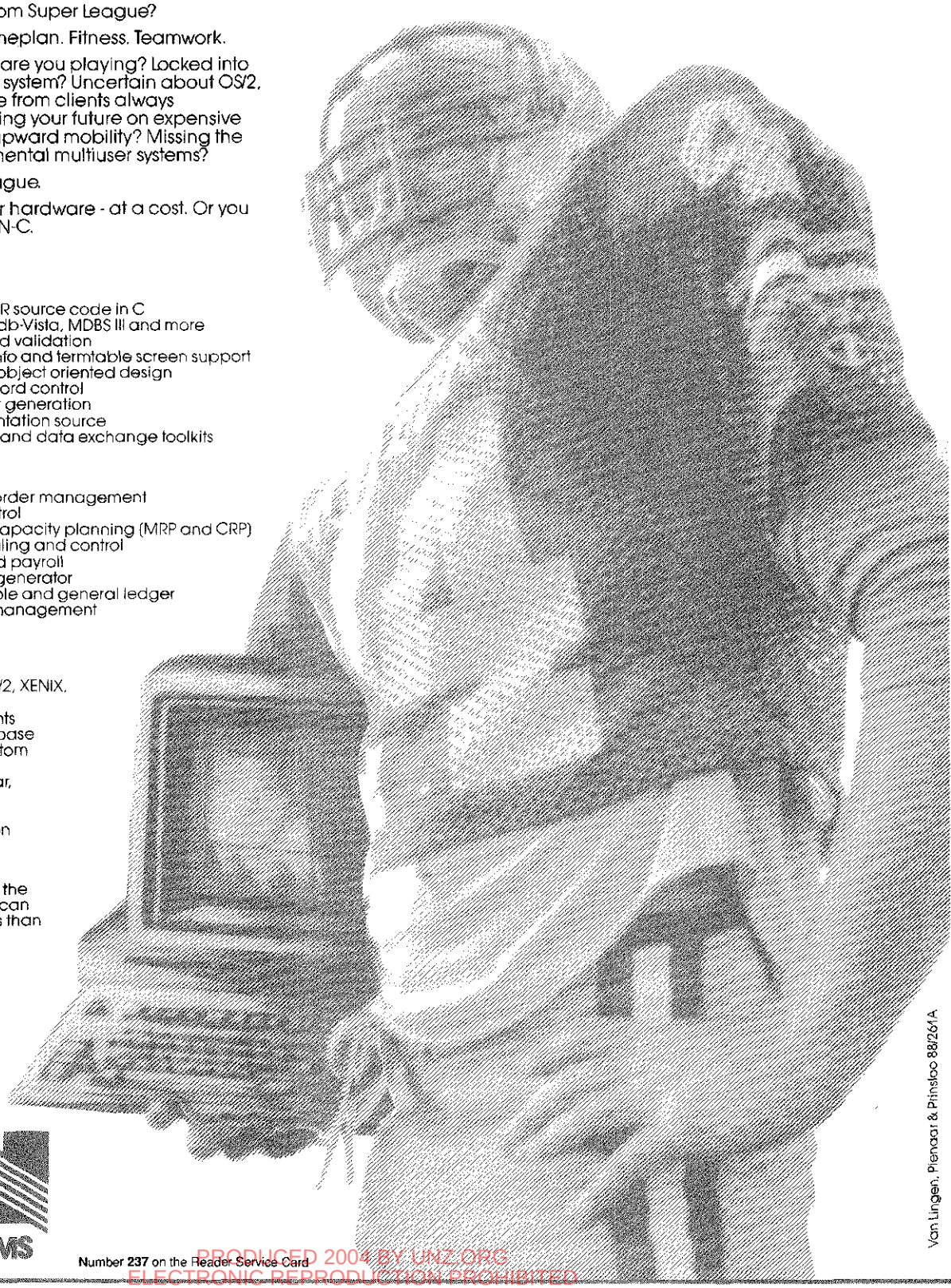
What seperates sand-lot from Super League?

Top Coaches. A better gameplan. Fitness. Teamwork.

So what kind of ball game are you playing? Locked into your proprietory database system? Uncertain about OS/2, UNIX, LAN's? Under pressure from clients always demanding more? Gambling your future on expensive 4GL's? No data sharing or upward mobility? Missing the massive growth in departmental multiuser systems?

That's no way to win the league.

You could standardise your hardware - at a cost. Or you could switch the play to XLN-C.



The skills

- Complete documented K & R source code in C
- Interfaces to c-tree, C-ISAM, db-Vista, MDBS III and more
- Sophisticated data entry and validation
- Video, BIOS, termcap, terminfo and termtable screen support
- Fast, portable and efficient object oriented design
- 1-2-3 style menus with password control
- WYSIWYG screen and report generation
- TEX typesetter and documentation source
- High level graphics, comms and data exchange toolkits
- LISP and Prolog AI system

The plays

- Sales, purchase and works order management
- Multi-location inventory control
- Distribution, materials and capacity planning (MRP and CRP)
- Critical path activity scheduling and control
- Personnel management and payroll
- SQL database application generator
- Accounts receivable, payable and general ledger
- Maintenance and service management
- Fixed asset management

The touchdowns

- No runtime royalties
- Identical source for DOS, OS/2, XENIX, UNIX and VMS
- Portable to 60 + environments
- Upwardly compatible database
- Flexible source code for custom applications
- Grow with standard, modular, integrated object design
- Fast, portable, multiuser, multitasking business solution
- Unlimited technical support

Gate money

$395 gets you in the gates with the module of your choice: or you can cover the whole season for less than the price of a PC.

XLN turns you into a master strategist: while C turns your cowboys into Cowboys.

EXCELENT SYSTEMS

Number **237** on the Reader Service Card.

Van Lingen, Prenaar & Prinsloo 88/261A

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SMKgen ($49.95), their makefile generation utility. These tools are best used together although Seidel Make (SMK) can be used alone. SMK's design differs substantially from Unix make's. SMK uses a different syntax—their Dependency Definition Language (DDL)—for make dependency files. Although similar in concept to Unix make's, Seidel keywords generally begin with #. DDL is generally easy to read and understand. SMK greatly extends the dependency language concept, supporting include files of constants and macros, conditional expressions similar to the C preprocessor's, a form of regular expressions for filename matching, filters for filename translation, block and line comments, epilog and prolog instructions, and many more features.

Part of the effort when using a make facility is creating the dependency file and keeping it up to date. SMKgen solves this problem. This powerful utility generates a complete make dependency file from either the command line or a very simple input file. To generate a makefile for a series of C modules in a subdirectory requires only a one-line command (SMKgen target *.c). SMKgen parses the source files for header (include) files and creates a dependency file in DDL along with a linker command file. The parsing scheme for header files is user controlled, and makefiles can be generated as easily for BASIC, FORTRAN, Pascal, and other languages as for C. The generated DDL file makes extensive use of SMK macro facilities and include files. The target compiler can be selected by changing a makefile constant, setting an environment variable, or via the command line when invoking SMKgen or SMK. Macro files supporting several C, FORTRAN, and Pascal compilers along with several linkers, librarians, and an assembler are supplied. Files are included for Aztec, Ecosoft, Lattice, Microsoft, Lahey, and Phoenix C, and for Oregon Pascal and generic files.

SMK operation also differs from other make utilities'. SMK executes in two passes, first checking the make dependency file for syntax errors and then creating a sophisticated batch file that is then executed. Through this design, SMK makes available the full 640K (less DOS, of course) for executing the make file commands. An environment variable SMKLABEL is maintained and used to monitor successful progress of the make generation. If an error occurs during the batch file execution, the user can edit an offending source file and re-invoke the batch file. The batch file would skip all previous steps that completed successfully and continue from the error. In fact, the default SMK macro library supports three error recovery techniques: (1) abort—fix errors and resume from where you left off by re-executing the batch file; (2) autoedit—automatically invoke your text editor on the source file containing compilation errors, and upon exiting the editor, the make resumes automatically; and (3) batch—where all error messages are logged to a file.

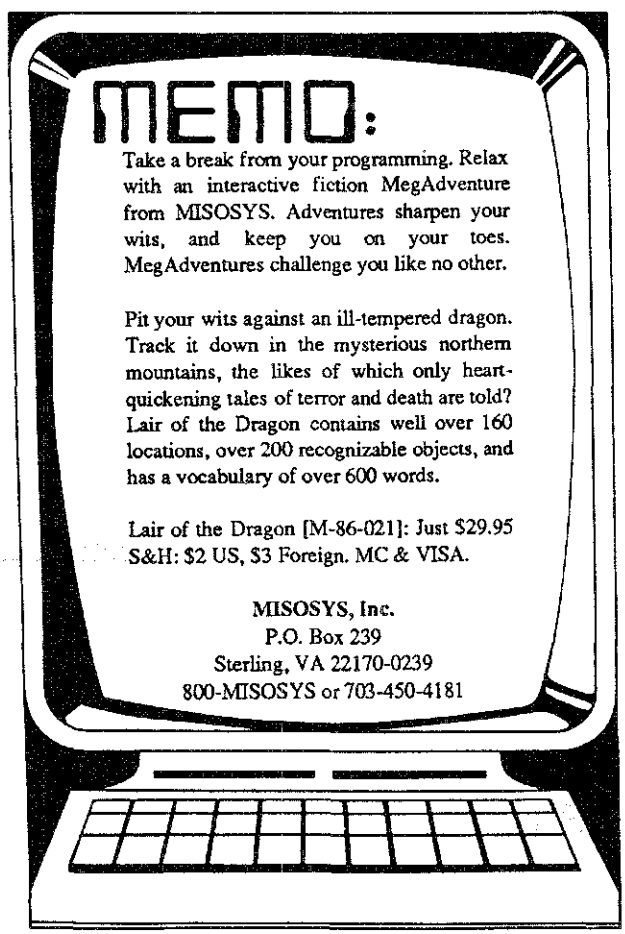
Seidel is porting these same utilities to 386 Unix and VAX VMS. SMK also supports their revision control program. The programs comes with solid manuals (SMK—114 pages, SMKgen—80 pages) with indices. I found these programs easy to use and very powerful.

Text Editors and Formatters

MKS SoftQuad Publishing Software
Mortice Kern Systems, Inc.
35 King Street North
Waterloo, Ontario
Canada N2J 2W9
(519) 884-2251

AT&T has been farming out some of its research and development efforts recently, especially in the area of Unix software. To that end, they've searched out companies with products and reputations most likely to enhance the AT&T image. SoftQuad, another competent Canadian Unix company in the Toronto area, was selected to rewrite the Documenter's Workbench (nroff and troff text formatting programs) for AT&T. Mortice Kern (MKS) has ported the entire SoftQuad Documenter's Workbench to DOS. The MKS SQPS ($495) is a rather impressive package. The base package comes with

Get To Know Your Programs Inside! and Out!
Now you can analyze your programs with unprecedented detail with Inside!, a new software package from Paradigm Systems.
Inside! allows you to examine the route your programs take through execution counts, minimum, maximum and total elapsed times and a count of how many times each source line executes—function by function—for popular Borland and Microsoft languages!
Now available
Inside! Turbo C
Inside! Turbo Pascal
Inside! Quick C
Inside! Quick Basic
Inside! Microsoft Pascal
Inside! Microsoft Fortran
Inside! Lattice C
Inside! Logitech Modula-2
PARADIGM SYSTEMS INCORPORATED
Paradigm Systems Incorporated
$75.00 Each
P.O. Box 152 Milford, MA 01757
To Place Orders
(800)537-5043
Product Support
(508)478-0499
Visa/Mastercard Accepted
Inside! is a trademark of Paradigm Systems Incorporated.


MEMO:
Take a break from your programming. Relax with an interactive fiction MegAdventure from MISOSYS. Adventures sharpen your wits, and keep you on your toes. MegAdventures challenge you like no other.
Pit your wits against an ill-tempered dragon. Track it down in the mysterious northern mountains, the likes of which only heart-quickening tales of terror and death are told? Lair of the Dragon contains well over 160 locations, over 200 recognizable objects, and has a vocabulary of over 600 words.
Lair of the Dragon [M-86-021]: Just $29.95
S&H: $2 US, $3 Foreign. MC & VISA.
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
800-MISOSYS or 703-450-4181

Number 195 on the Reader Service Card

Number 238 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOS versions of the AT&T text formatting, preprocessor, and macro package software along with generic drivers for printers and video displays. The base output drivers are Epson FX and LQ compatible dot matrix printers and the CRT using ANSI. Two additional driver packages are available for the HP laser printers and PostScript devices ($200 each).

SQPS represents a complete text formatting and typesetting system compatible with Unix. Troff and the SQPS package are not WYSIWYG programs like Ventura Publisher, but are batch formatting utilities. SQPS can display output to video display devices for preview purposes, but all formatting commands are entered into text files. The MKS SQPS software supports normal typesetting capabilities, including size and control of fonts and characters, text filling, text adjustment, centering, page control, hyphenation, kerned fonts, and font mapping. The strength of these programs is in the powerful macro packages designed to simplify the entry of tables and equations, create diagrams and view graphs, and manipulate text. Basic program operation consists of entering text formatting commands into a text file. These formatting commands may include using one or more of the macro packages. The troff program acts as a filter on this input file and produces an intermediate file as output. This intermediate file may then be previewed on-screen so that you can review the formatted document before you send it to the output device. Postprocessors convert this intermediate file to a specific output device like a laser printer. The drivers provided for dot matrix printers are text files, which may be modified to support other output devices.

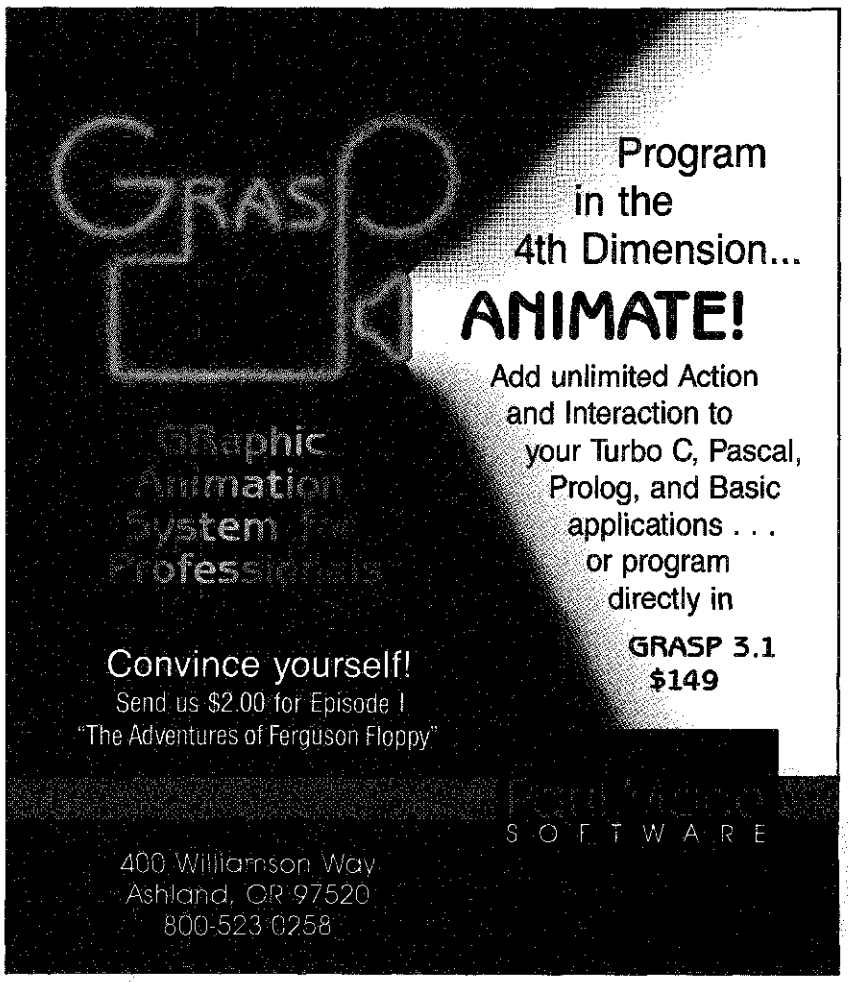
The base SQPS package comes compressed on 6 disks, and the Laserjet drivers add another 9 disks. I reviewed only preliminary copies of the manuals, which are in the Unix tradition—sparse. If you need nroff or troff compatible tools under DOS, this package may be the ticket.

SEE Editor
C Ware Corporation
945 Spring Street
Paso Robles, CA 93446
(805) 239-4620

While reviewing text editors, I was asked on occasion what editor I use. For most work, I've used the SEE editor that comes bundled with C Ware's DeSmet C compiler although I might change one of these days. This editor falls somewhere towards the other end of the editor spectrum from powerhouses like BRIEF and PI. SEE is small, less than 45K executable file, and uses about 100K of memory maximum. It is fast for small files (less than 60K) and supports only limited capability. Its modest size allows invoking it from other environments such as dBASE or FoxBase. I'm also used to it, which makes a difference. The limited command set doesn't take much time to learn. Much of my text editing is on small files—batch and make files. With a small quick editor, I can complete a change before a larger, more powerful editor would be finished setting up its environment. SEE can read and edit files ending either with carriage return-linefeed (DOS) or linefeed only (Unix), so file conversion isn't necessary. The editor can also be used on larger files since it supports a reasonable set of features.

The SEE editor as bundled with DeSmet C ($99) includes an executable version with default keystroke bindings, an object file for most of the editor, and a configurable C (for terminals) and equivalent 8086 assembly language source file (IBM PC) that contains all keyboard input and screen output primitives. You are free to reconfigure the keyboard input source to your liking, link, and create a custom version. The editor source code in C ($100) is also available for personal use. SEE features both an edit, or visual, mode (either insert or overtype) and a command-line mode that supports limited macro capabilities (repetition, replace with query, binding macros to function keys, and configuration). SEE can edit up to two files using horizontal split-screen windows. A single keystroke expands a window to full-screen mode. Features are configurable from within the editor, including the size of the two windows, tab expansion spacing, whether auto-indenting is enabled, whether find/replace commands should be case sensitive, and creating, loading, and saving macro files.

The editor is virtual and can handle file sizes up to the DOS partition limits. A single buffer is provided for deleting and moving text. Delete and block commands are limited and somewhat awkward. To move a block of text, you must delete it (place it in a buffer) and then copy it to its destination. You can do block deletion and copying only after escaping to the command mode; then you must re-enter the edit mode. To duplicate a block, you must copy it to the buffer, then copy it to its final destination. You can make multiple copies of this buffer, though. This delete buffer provides one level of undo. Moving or copying text between windows uses the same buffer. This single buffer is virtual—using 4K of memory and otherwise spilled to disk so that the size of block moves and undeletes is not limited. You can also write marked blocks to a disk file and read in other files at the cursor position. After editing, a file may be backed up or overwritten (in SEE jargon, "updated"). File overwriting, although not recommended for source code, is convenient for small batch files—you don't have to constantly erase backup files. You can place up to four markers (tags) in a file and jump to a line number or a tag. The editor supports an escape to DOS and issuing a compile command. SEE has obvious deficien-



Number 80 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

cies, but the basic features of a solid editor are there. Mike Ouye, the author of SEE, worked for Intel before venturing out on his own. Aedit, the editor Intel sells with their DOS development tools, has an obvious SEE feel (my guess is that Mike was involved in Aedit's development at Intel). Source code to SEE is distributed on a single disk without documentation.

Programming Utilities

C:LINES/C:TREE
SoftRex
4807 Bethesda Ave, Suite 287
Bethesda, Maryland 20814
(301) 881-8274

In a previous column (*PJ* 6.3), I praised the C tools from MMC AD Systems, particularly their CFLOW utility. I've found another set of excellent utilities for C from SoftRex. Newly released Version 2.0 of C:LINES/C:TREE ($79.95) is an excellent package of source code listing and analysis tools for C programmers. The package provides four main functions: (1) a source code reformatter; (2) an outlined listing of all logical blocks of a C program or module; (2) a detailed cross-reference of C symbols and how they are used; and (4) a hierarchy tree that shows the relationship of various functions—a flow chart. The tools consist of three main programs that work together: CLINES, CTREE, and CFORMAT.

What distinguishes this package is that C:LINES/C:TREE V2.0 has acquired full preprocessing capabilities and parses C files just as your favorite compiler does. Header files are automatically scanned and listed under option control. The cross-reference and the flow tree reports reflect the contents of code after all macros have been expanded. Code contained within false conditional blocks can be ignored. The parser provides support for all proposed ANSI C keywords and ANSI trigraphs. Special DOS C keywords like **far**, **cdecl**, and **_ds** can be entered into a configuration file and enabled as legitimate C keywords. Sample support files are supplied for compilers from Datalight, DeSmet, Borland, Microsoft, and Watcom. Printer controls are handled in a similar way, and some common dot matrix printer files are included.

The programs support a very long list of command-line options, which may also be taken from a file. Unused global symbols that are never used can be reported. The listing utility will mark locations of all return statements on the outline so that all exits from a procedure can be readily identified. Break and continue statements can also be noted. CLINES can breeze through large complex C projects with defines, macros, and header files and generate a clear flow chart or a structured listing for analysis. In short, no other product can quite equal it.

Multitasking

DESQview 2.0 API Tools
Quarterdeck Office Systems
150 Pico Boulevard
Santa Monica, California 90405
(213) 392-9851

It's really quite ironic—a product based on the now defunct IBM TopView is not only successful, but it's competing against IBM's second try at PC multitasking, OS/2. DESQview ($129.95) is what IBM's TopView should have been. For reasonably well behaved applications, the mature DESQview provides multitasking and text windowing under DOS. Quarterdeck also sells QEMM-386 ($59.95), an expanded memory manager for 80386 compatibles with LIM 4.0, and a similar product for PS/2 Models 50 and 60. Last December, Quarterdeck and Phar Lap (386 development tools) jointly authored the Virtual Control Program Interface (VCPI), a protocol for cooperation among native mode 80386 programs that's designed to prevent the TSR conflicts that go on under DOS. The latest versions of DESQview (2.2) and QEMM-386 (4.1) support the VCPI.

This Spring, Quarterdeck introduced the DESQview 2.0 API (Application Programming Interface) tools, a set of libraries and documentation to take advantage of DESQview 2.0 features. The toolkits currently available are the API Reference ($59.95) and the API C Library ($199.95), both with solid manuals. The API is a set of subroutines and functions through which a program can interact with and control DESQview. The API covers multitasking, windowing, intertask communication, data transfer, memory management, customization, and portability issues. The API also supports mailboxes, menu building, on-line help, and using shared program code. The API Reference describes DESQview API calls in assembly language and provides a disk of assembler macros. The API uses both a Direct Call interface using 8086 registers and a Send interface that pushes paramaters on the stack. Interrupt 15H is used primarily for the API. The DESQview environment is well documented, including context switching, memory allocation, video issues, the keyboard, DOS, direct I/O, and hardware and software interrupts. The API C Library provides a C language interface for the entire set of API functions. The C Library Toolkit comes bundled with the API Reference. C compilers from Lattice, MetaWare, Microsoft, and Borland are currently supported for all memory models. The source code as well as the object libraries are included on disk.

Although programs may be quite easily written to be DESQview aware (see *PJ* 6.4, "Windows within Windows"), the API allows applications to take advantage of more advanced DESQview features. This allows the construction of a package of cooperating applications that share information. During a Summer Quarterdeck conference for developers, some impressive custom applications using the API were demonstrated.

Corrections

Ray Duncan pointed out that the BRIEF macro language (*PJ* 6.5) is *prefix* (like LISP), not *postfix reverse Polish* (like Forth).

The correct address for the Iliad Group (the PI editor, *PJ* 6.5) is 2697 Union Street, San Francisco, CA 94123. ♦

PJ Code Is Available!

On disk from

Programmer's Journal
(503) 344-8151
See ad on page 88

On BBS from

Mnematics Videotex
P.O. Box 19
Sparkhill, NY 10976-0019
800-322-3633
914-365-0184 in NY
PJ readers are eligible for free subscriptions to Mnematics Videotex.

Turbo Source Search
P.O. Box 876
Scituate, MA 02066
Voice (617) 545-6677
Data (617) 545-9131
All *Programmer's Journal* subscribers are pre-registered in the TSS callers' log. See below.

FIDO/OPUS Network
Blazer Net 152 Node 9
(503) 343-9724

Generation Five BBS
Voice 301-495-6690
Data 301-495-2932
Accessible via PC Pursuit, Washington, D.C. From Files submenu, select P for Personal Downloads.

PCS-49 on CompuServe
Section 9
P.O. Box 239
Sterling, VA 22170

All *Programmer's Journal* subscribers are pre-registered in the Turbo Source Search callers' log. To download *PJ* code,

Have your modem call (617) 545-9131. At the prompt, type in your first and last names (no middle initial, no punctuation). At the next prompt, type in your password (the first four or five numbers of the subscription ID number on your *PJ* mailing label).

At the command prompt, type J (for Join); then type 3 for the *PJ* Conference.

Once in the *PJ* Conference, type F (for File Listings) to view the sub-directory menu. Type in the number of the sub-directory you want to see and note the name(a) of the file(s) you want to download.

Type D (for Download). At the prompt, type in the name of the file(s) you want to download.

For help, type H and then the key letter from the Main Menu. For further help, leave a message for SYSOP.

To unarchive the files you download, use file PK36.EXE (Phil Katz's excellent archive tools). We recommend that you use ARCM240.ARC (John Newlin's ARCMASTER) in conjunction with PK36.EXE.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EDITOR $50

SEE™ is an easy to use, full screen, full function text editor designed for both the naive user and the experienced programmer.

SEE handles files larger than the amount of free memory available and can edit two files simultaneously.

Since SEE is easy to use and is easily embedded in hardware products, OEMs find it to be their editor of choice. Quantity discounts available.

BLINDINGLY FAST, MODESTLY EXCELLENT AND ABSURDLY CHEAP. SEE, The Editor. Psst . . . Only $49 more buys you a C compiler too. DeSmet C at $99 includes SEE.

C Ware Corporation
945 Spring St., Suite 14
Paso Robles, CA 93446
805-239-4620 **RS#38**

OPTASM, THE FASTEST MACRO ASSEMBLER AVAILABLE

Processing 75M lines/minute, OPTASM is a macro assembler (source code compatible with MASM) that is loaded with features:

- Works with Borland and Microsoft languages
- Demo: FREE from SLR Systems's BBS (412-282-2799) or send $10 to SLR Systems, and we'll send you a disk
- Now available with CodeView support

OPTASM $125 (list) CALL TO ORDER or FOR MORE INFO: 800-833-3061

SLR Systems
1622 North Main
Butler, PA 16001
412-282-0864 **RS# 165**

NATURAL LANGUAGE FOR YOUR dBASE APPLICATIONS

What would YOUR customers pay for a natural language query interface for their dBASE/FoxBase applications? With ELLIE you can build memory-resident ad hoc query capabilities into any (interpreted mode/non-compiled) dBASE language system. Rated "Excellent" by USER (University Software Evaluation Report) for both ease-of-setup and user-friendliness (3.6 out of a 4 overall rating). Full-featured, with transformational grammar processing, automated installation/dictionary-building, support for synonyms, antonyms, phrase definition, unit conversions, macro handling and more. Your cost: $119 plus $50 per site/client runtime license fee. Your profit opportunity: UNLIMITTED. And there's a full 30-day money back guarantee.

ELF Software Company
210 West 101st Street
New York, NY 10025
212-316-9078 **RS# 113**

UNDERSTANDING C VIDEO TAPE

Get guaranteed results with this three volume video course! For expert or novice—these tapes (total 6 hours) cover ANSI C for today's professional programmer. Loaded with examples, these tapes cover it all! Handy reference book included.

$59.95 postpaid. Add appropriate tax in TX. Send check or money order to:

AUTOMATION EDUCATION
2309 Royce Drive
Arlington, TX 76016-1232
817-496-4125 **RS# 207**

BASIC to C

BAS C v3.2 accepts BASICA, Quick BASIC, CBASIC86, or most other BASICs by adding new statements, functions(Language Extender). Generates structured, indented, scoped MS/Turbo/Lattice/Aztec C. All Memory model, compile option supported. Runs on MS-DOS, XENIX, UNIX. Demo disk. From $199.

GOTOLESS CONVERSION
P.O. Box 835910
Richardson, TX 75083
214-404-1404 **RS# 127**

SEE CURRENT FILE ACTIVITY

WATCH I-O is a pop-up utility that displays files open, DX:DS of open and last read/write, last read/write counts, last exec, last file opened, last file-not-found. User can select hot key, pop-up for file opens, pop-up for file-not found, track file opens on printer.

WATCH I-O is only $49.95. With anti-virus option $69.95. VISA/MC.

Quantum Publishing
23860 Miles Road
Cleveland, OH 44128
800-342-6237 **RS# 201**

PL/M TO C TRANSLATOR

LISP-based translator for major assistance in translating from INTEL PL/M to C language. Human readability a high priority. Sample listings available. Runs under MS-DOS 2 and up. Ports possible for VAX with VMS or BSD 4.2.

$5000 site license includes 3 months of telephone support.

Frank Korzeniewski Consulting
1564-A Fitzgerald Drive #137
Pinole, CA 94564
415-799-1819 **RS# 209**

INTELLIGENT REVERSE-ASSEMBLER

MASTER*KEY is an intelligent self-documenting MS-DOS disassembler that returns fully documented formatted assembly language source code (MASM 5.0 compatible) from ANY binary file of ANY size. It works from the command line or interactively with menus. Selective partial output speeds through a program file keeping listings and source files down to a manageable size. MASTER*KEY's cross-reference locates all items by memory location in the program file.

$79.95 + 2.50 shipping. Add 6 1/2% sales tax in CA.

Sharpe Systems Corporation
2320 E. Street, Dept. 1610
La Verne, CA 91750
RS# 175

AT LAST! A C SOURCE CODE ANALYSIS TOOL WITH FULL PREPROCESSING CAPABILITY

C:LINES/C:TREE features

- ANSI draft compatible preprocessor
- Outlined listings showing flow of control within procedures
- Tree diagrams showing flow of control between procedures
- Source code reformatter

$79.95, shipping $3 domestic, $15 overseas. MD residents add 5% sales tax. Requires PC compatible, 256K.

SoftRex
4807 Bethesda Ave. #287
Bethesda, MD 20814
301-881-8274 **RS# 185**

HOT NEW UTILITY!

EXPERT KEY is a RAM-resident keyboard enhancement utility that learns words and phrases and later recognizes them after a few keystrokes. Using AI and Expert Systems techniques, it finds and pops up a smart prediction; you either tap the EXPERT KEY to complete the word or keep typing. Learns words, phrases, lines & editing sequences from the keyboard, screen or your personal dictionary. Highly configurable. Runs with your editor and applications. For PC/XT/AT/386.

$65 (Reg. $95) + $3 S&H, 30 day guar.

Newell Technologies, Inc.
27758 Santa Margarita Pwky. #330D
Mission Viejo, CA 92691
714-583-7848 **RS# 203**

Computer & Technical
***** BOOKS *****
INFOSOURCE
International
PROGRAMMER'S PC SOURCEBOOK **$24.95**
TURBO C DEVELOPER'S LIBRARY **$24.95**
Box 238, Dept. PJ 108
Barryville, NY 12719–0238
1–800–327–7970

Number **217** on the Reader Service

WITH VM/386 YOU CAN DO TWO THINGS AT ONCE EASILY:

- Download E-Mail while updating a spreadsheet.
- Plot one CAD drawing while creating a new one.
- Perform 3270 emulation while running Symphony®.
- Update budgets while you print labels off a database.

And it's only $245. Call IGC today for more information.

IGC
4800 Great America Parkway
Santa Clara, CA 95054-1221
408-986-8373
Toll-Free: 800-458-9108
FAX: 408-986-1431 **RS# 169**

PERSONAL MEASURE

measures PC performance. Use to tune system, evaluate products, do benchmarks. TSR less than *10K* & self removing. Reports w/ menus & graphics. DOS 2.0+; CGA, EGA, VGA, Hercules graphics; FX-80, Proprinter, LaserJet. *$69.95*, + $2 S&H, $5 for 3 1/2" media, 5% tax for MA residents. Sensible 30-day money back guarantee.

VISA MC

Spirit of Performance, Inc.
73 Westcott Road
Harvard, MA 01451
☎ *508-456-3889*
MCI Mail 357-1400

Number **197** on the Reader Service

NOW AVAILABLE!
Programmer's Journal DiskScription—
See page 8 for details

All the source code from each issue plus all the code we didn't have room to print—mailed direct to you on a 360K diskette. Become a charter subscriber today for only $29.95 per year.

PJ ISSUE SPECIAL!
Back Issue *PJ* Code Disks for $5.00—See page 88 for details

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service	Advertiser Name	Page Number
227	Abraxis Software, Inc.	28
220	Aesoft Corporation	24
180	A.I. Architects, Inc.	77
215	American Cosmotron, The	21
184	American Cybernetics	34
162	Anji Software, Inc.	87
86	Aspen Systems	76
30	Austin Code Works	52
207	Automation Education	95
51	Az-Tech Software, Inc.	68
38	C Ware Corporation	95
231	Cobalt Blue	54
56	CompuView Products, Inc.	13
*	Creative Programming Consultants	30
229	Disk Software, Inc.	80
8	Ecosoft, Inc.	16
113	ELF Software Co.	95
9	Entelekon	55
206	Flambeaux Software	44
118	Fleming Software	28
209	Frank Korzeniewski Consulting	95
135	Gilmore Systems	43
*	Gimpel Software	47
127	Gotoless Conversion	95
204	Guidelines Software, Inc.	27
42	Harvard Softworks	59
122	Hauppauge Computer Works, Inc.	C4
154	HavenTree Software LTD	20
169	IGC	95
217	Infosource International	95
239	INTEK Integration Technologies	15
226	Ithaca Street Software, Inc.	43
202	Javert, Inc.	80
211	Kadak Products LTD.	44
155	Komputerwerk	41
63	Laboratory Microsystems, Inc.	61
44	Lahey Computer Systems, Inc.	84
212	Magna Carta Software	51
233	Mansfield Software Group	15
223	Marietta Systems	11
213	Max Software Consultants	79
230	McSoft, Inc.	71
235	Meridian Software Systems, Inc.	66
143	Metagraphics Software Corporation	85
45	MetaWare, Inc.	1
214	Micro Interfaces Corporation	63
232	Microsoft Corporation –Human Resources	9
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

Reader Service	Advertiser Name	Page Number
234	Microsoft Press	25
104	MicroWay	75
121	MISOSYS Inc.	90
238	MISOSYS Inc.	92
94	MMC AD Systems	10
109	NanoSoft	15
203	Newell Technologies, Inc.	95
160	Nu-mega Technologies	3
16	Opt-Tech Data Processing	79
195	Paradigm Systems	92
145	Pathfinder Associates	86
80	Paul Mace Software	93
208	PC Tech, Inc.	39
216	Peacock Systems	70
6	Periscope Company, The	12
26	Polytron Corporation	31
*	*Programmer's Journal* Disk Subscription	8
*	*Programmer's Journal* Mail Order	88
200	Programmer's Paradise	7
228	Protech Marketing, Inc.	11
97	Qualstar Corporation	76
201	Quantum Publishing	95
147	Quarterdeck Office Systems	C2
222	Rainbow Technologies	69
20	Rational Systems, Inc.	49
198	Research Group, The	18
148	SemWare	63
175	Sharpe Systems Corporation	95
165	SLR Systems	95
185	SoftRex	95
236	Soft Source	26
98	Software Family, The	89
221	Software Security	5
110	Softway, Inc.	36
197	Spirit of Performance, Inc.	95
158	Summit Information Systems, Inc.	14
85	Sunny Hill Software	17
171	Systemat Corporation	6
166	Tool Makers, The	84
25	TurboPower Software	82
35	Turbo Source Search	79
125	V-Communications	37
196	West Chester Group	36
100	Western Wares	11
237	XLN Software	91
123	Xenosoft	68
182	Zortech, Inc.	C3
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine.

ADVERTISE IN

Programmer's Journal

Jan/Feb/89 Issue

Volume 7.1

Space Reservation Deadline:

November 4, 1988

Materials Due:

November 18, 1988

Call Programmer's Journal Ad Department to reserve your space.

(503) 747-0800

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

See us at
COMDEX/Fall '88
November 14–18, 1988
Riviera Hotel
Las Vegas, Nevada

INTRODUCTORY PRIC
$99.95
Regular price $149.9

WALTER BRIGHT – Programmer

"This is a revolutionary product which reflects the fact that Walter Bright is one of the world's best programmers."
Andy Redfern, Personal Computer World

"I find it amazing that Walter continues to find ways to improve his already excellent product."
Scott Robert Ladd, Micro Cornucopia

FROM ZORTECH INC.

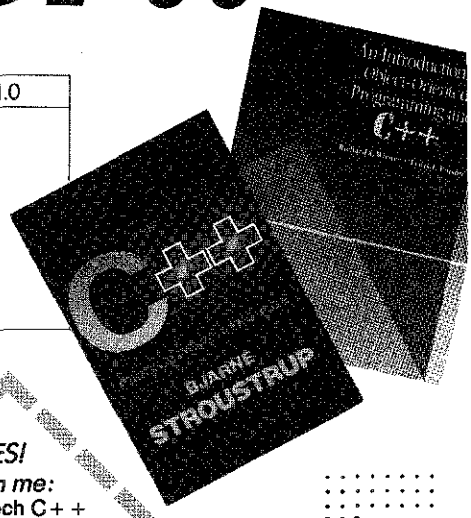
C is great!
C++ is better!!

Performance benchmarks

Benchmark	Zortech C	Zortech C++	Turbo C 1.5	Quick C 1.0
Sieve	20.49	20.54	23.62	22.72
Rsieve	20.49	20.54	23.62	22.03
Integer	1.32	1.38	6.31	6.49
Float	0.17	0.22	52.29	51.03
*Float	32.73	37.74	52.39	51.63
Pointer	17.91	17.96	17.13	16.87
Rpointer	17.79	17.91	17.14	16.64
Loop	3.90	3.90	3.90	3.90
Optimize	0.49	0.60	8.46	8.79

Benchmarks were run on an 80286 based IBM compatible at 6Mhz with no 8087.

*The float benchmark was re-run without optimization.



Zortech C++ is the first native-code C++ compiler for MS-DOS compatible machines. This means you don't need to spend a lot of money on a C compiler. In fact, Zortech C++ comes with a C++ compiler, a C compiler, a linker, a librarian, a fully integrated editing environment, context sensitive help and the fastest graphics library you've ever seen!

With Zortech C++, you can mix and match code in C++, C and assembler and link them all together with a single command!

Zortech C++ comes with a complete C++ and C library, including compatibility with the ANSI C draft. Unique to Zortech C++ is support for mice, fast screen writes, re-entrant floating point and speaker support.

Now you can move your Microsoft and Turbo C code over to Zortech C++ – we've included compatible library functions! Also, we have made Zortech C++ compatible with Microsoft's Codeview debugger!

Enter the world of Object Oriented Programming (oops) with Zortech C++!

Sure, C is great – but C++ is better!

NOW AVAILABLE...
Std. Library Source Code
Only $49.95 * (with C++ Purchase)

YES!
Rush me:
☐ Zortech C++ 99.95*
☐ Library Source $49.95*
☐ Intro to OOP and C++ $27.95*
☐ The C++ Programming Language $29.95
VISA/MC/COD CHECK ACCEPTED

ZORTECH

Name.....
Address.....
..... *Phone*.....
VISA or MC#..... *Exp. date*.....

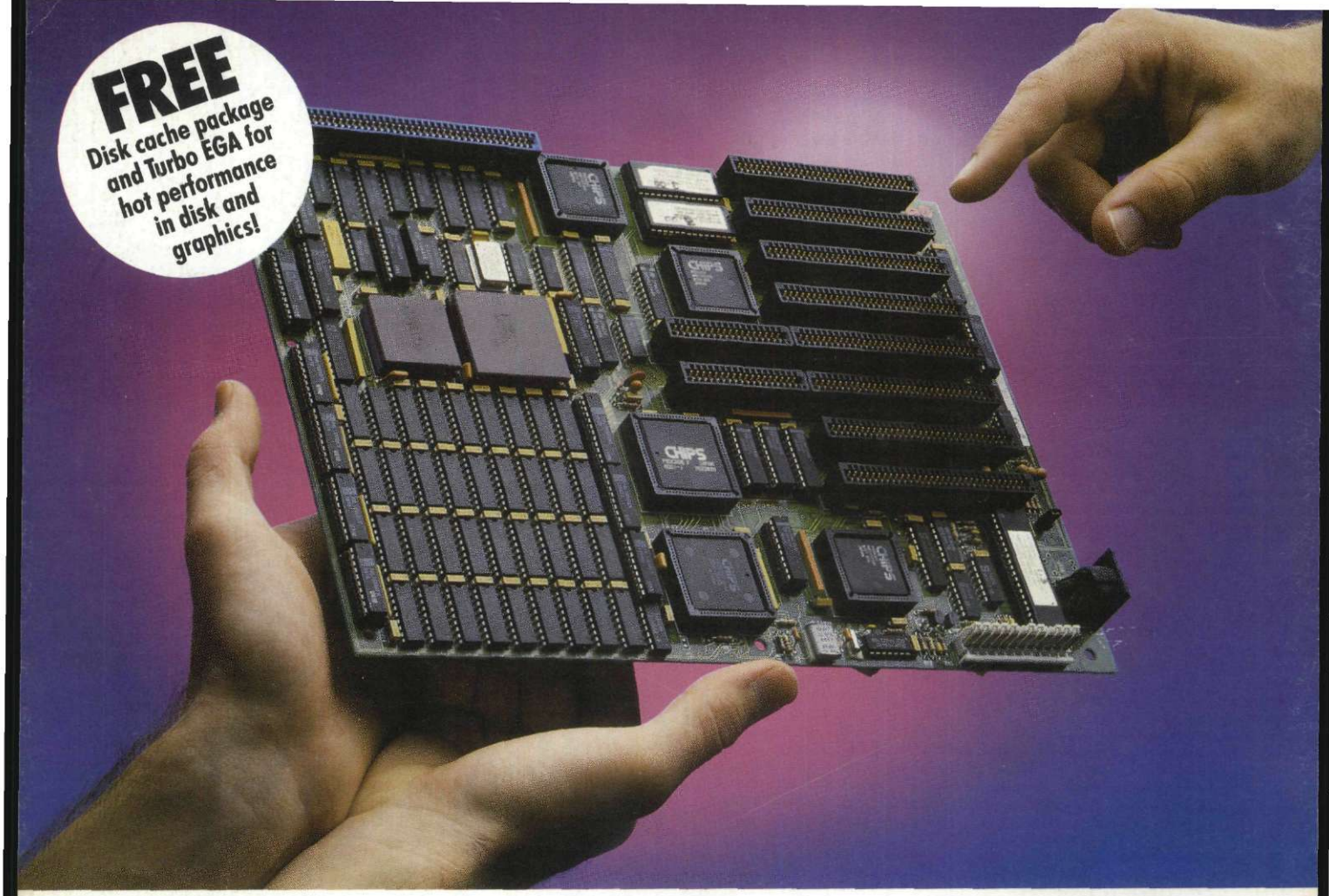
To: ZORTECH INC. 366 Massachusetts Ave., Arlington, MA 02174. Tel: 617-646-6703. Fax: 617-643-796

CALL THE ORDER HOTLINE 1-800-848-8408

Number 182 on the Reader Service Card @ 11/12

* Please add $5.05 shipping to your order (UPS Ground)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



CONVERT YOUR PC, XT OR AT INTO A HIGHER FORM OF LIFE!

386 MOTHERBOARDS FOR 386 SPEED.

Don't let your PC give up the ghost — Hauppauge has arrived with a new spark of life: the 386 MotherBoard.™ Far more advanced than an accelerator card, our line of MotherBoards grace your PC, PC/XT, PC/AT or compatible with speeds equal to the IBM PS2 Model 80. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that will make your PC glad to be alive.

UNIX/386 Compatible. To ensure a long, fruitful life, our 386 MotherBoards are compatible with the PC/AT (BIOS and I/O) — so you can run the new generation of DOS, OS/2. Our Boards also run Windows/386, UNIX V, PC-MOS/386, and XENIX/386. For *more* power, you'll find 16-bit expansion slots that accommodate the latest I/O expansion cards. Plus, we can add up to 15 Megabytes of extended memory. No 386 accelerator card can go over 2 Megabytes or give you so much versatility. **Only our 386 MotherBoards give your PC s an unlimited future!**

The Critics Applaud! *PC Magazine* awarded our Board "The Editor's Choice" for 386 Replacement Boards. *PC World* called it "the Upgrade Product of the Year."

Technical Features ▪ 16 or 20 MHz 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Six 8-bit expansion slots ▪ Two 16-bit expansion slots; four on 386 MotherBoard for AT ▪ One 32-bit expansion slot for up to 15 Megabytes of high speed memory ▪ Battery-powered clock calendar ▪ 2.9 MIPS at 16 MHz, 3.9 MIPS at 20 MHz.

386 MotherBoard for the PC or PC/XT	$1695
386 MotherBoard for the PC/AT	$1795
386 MotherBoard/20 MHz for PC/AT	$2195
32-bit RAM Board (2 Mbyte installed; up to 10 Mbytes)	$1195
16 MHz 80387 math coprocessor	$695
16-bit combination hard disk/floppy disk controller	$245

Prices subject to change.

For more information on our easy-to-install MotherBoard, call **1 (800) 443-6284**. In New York: **(516) 434-1600**. All our products are available at your local computer dealer!

Hauppauge Computer Works, Inc.
175 Commerce Drive,
Hauppauge, New York 11788.

Hauppauge!

Trademarks: IBM PC, XT, AT, PS2 and OS/2: IBM.
Intel 386: Intel Corp.; Windows/386: Microsoft Corp.

Number **122** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED